

Encyclopaedia

Author and Editor

Elzein Salah Elzein

IEEE Member

- Papers in Algorithms

IEEE Publisher, UK



Article

Convolutional Neural Network-Based Digital Diagnostic Tool for the Identification of Psychosomatic Illnesses

Marta Narigina , Andrejs Romanovs * and Yuri Merkuryev

Department of Modelling and Simulation, Riga Technical University, 6A Kipsalas Street, LV-1048 Riga, Latvia; marta.narigina@rtu.lv (M.N.); jurijs.merkurjevs@rtu.lv (Y.M.)

* Correspondence: andrejs.romanovs@rtu.lv

Abstract: This paper appraises convolutional neural network (CNN) models' capabilities in emotion detection from facial expressions, seeking to aid the diagnosis of psychosomatic illnesses, typically made in clinical setups. Using the FER-2013 dataset, two CNN models were designed to detect six emotions with 64% accuracy—although not evenly distributed; they demonstrated higher effectiveness in identifying “happy” and “surprise.” The assessment was performed through several performance metrics—accuracy, precision, recall, and F1-scores—besides further validation with an additional simulated clinical environment for practicality checks. Despite showing promising levels for future use, this investigation highlights the need for extensive validation studies in clinical settings. This research underscores AI's potential value as an adjunct to traditional diagnostic approaches while focusing on wider scope (broader datasets) plus focus (multimodal integration) areas to be considered among recommendations in forthcoming studies. This study underscores the importance of CNN models in developing psychosomatic diagnostics and promoting future development based on ethics and patient care.

Keywords: convolutional neural networks; artificial intelligence; emotion recognition; psychosomatic illnesses; facial expression analysis; real-time emotion detection



Citation: Narigina, M.; Romanovs, A.; Merkuryev, Y. Convolutional Neural Network-Based Digital Diagnostic Tool for the Identification of Psychosomatic Illnesses. *Algorithms* **2024**, *17*, 329. <https://doi.org/10.3390/a17080329>

Academic Editor: Gabriella Trucco

Received: 29 June 2024

Revised: 23 July 2024

Accepted: 23 July 2024

Published: 30 July 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The accurate diagnosis of psychosomatic illnesses presents a significant challenge within modern healthcare. Psychosomatic illnesses, where psychological factors substantially influence physiological functions, often manifest symptoms similar to those of purely physical ailments. This overlap can lead to misdiagnoses and consequently, inappropriate treatment strategies [1–3]. With the rising integration of artificial intelligence in diagnostic processes, there is a burgeoning opportunity to enhance diagnostic accuracy using technological tools [4,5].

Recent advancements in machine learning, particularly convolutional neural networks (CNNs), have shown promising results in various medical applications, including image-based diagnostics and behavioral predictions [6]. The ability of this technology to analyze and interpret subtle patterns in facial expressions offers a novel approach to identifying emotional cues that are indicative of psychosomatic illnesses [7].

This study aims to bridge the gap between psychological assessments and traditional diagnostic methods by integrating a CNN-based tool to analyze facial expressions, providing an auxiliary means of diagnosing psychosomatic illnesses. This research revolves around several pivotal questions including the following:

1. What are the current limitations in diagnosing psychosomatic illnesses using traditional methods [8]?
2. Can a CNN model trained to recognize emotional expressions improve the diagnostic accuracy for psychosomatic illnesses [9]?

3. How effectively can such a model be integrated into clinical workflows to support medical professionals without replacing their expert judgment?

By addressing these questions, this article explores the feasibility and effectiveness of employing a CNN-based diagnostic aid in clinical settings, thereby enhancing the overall accuracy and efficiency of psychosomatic illness diagnosis [10].

The sections are meticulously structured to provide a detailed analysis of the various aspects involved in constructing CNN-based diagnostic tools. Every part is designed to delve deep into each stage of the development process.

Section 2, Related Works, examines different works that have used deep learning methods to enhance face emotion recognition. These techniques have been evaluated using the FER-2013 dataset, and they have shown notable enhancements in emotion recognition accuracies.

Section 3, Methodology, details the dataset, CNN architecture, training process, preprocessing steps, and multimodal integration of facial analysis and pulse data for predicting psychological disorders.

Section 4, Evaluation of CNN Models, assesses the reliability and effectiveness of CNN models using various performance metrics and cross-validation techniques in simulated clinical environments.

Section 5, Results and Analysis, presents performance metrics of CNN models, strengths, areas needing improvement, and statistical validation through confusion matrices and accuracy graphs.

Section 6, Discussion, interprets the significance of the findings, potential biases, limitations, ethical considerations, and practical implications of integrating AI tools in healthcare.

Section 7, Conclusions, summarizes the key findings and recommends enlarging datasets, conducting real-world validations, and combining multimodal data to enhance diagnostic capabilities, emphasizing ethical considerations and AI's potential impact on improving patient care.

Every section is created to lead from the previous one—all towards building a full-bodied discussion that underscores the great potential CNNs hold in changing the ways diagnoses are performed in healthcare setups.

2. Related Works

Recent advancements in deep learning and computer vision have significantly impacted the field of Facial Emotion Recognition (FER). This section reviews various studies that have employed deep learning techniques to enhance the performance of FER systems, particularly those evaluated using the FER-2013 dataset.

Proposed in reference [11], Georgescu et al. presented an approach where features learned automatically by a number of CNN architectures are merged with handcrafted features through the bag-of-visual-words (BOVW) model. Emotion classification is achieved via this hybrid methodology, adopting a local learning framework and depicting a better performance in terms of accuracy as a result of the combined use of both automatic and handcrafted feature sets.

Pecoraro et al. [12] developed a Local Multi-Head Channel (LHC) self-attention module referred to as a complementary component to the standard convolutional neural network (CNN). This mechanism avoids the generic actions of attention and concentrates instead on those parts of the face that are key actors in emotional expression. It was demonstrated that if you introduce LHC modules, you can boost the accuracy of facial expression recognition by localizing critical areas on the face.

Fard et al. [13] introduced Adaptive Correlation (Ad-Corre) Loss, which consists of three elements including Feature Discriminator, Mean Discriminator, and Embedding Discriminator. Ad-Corre Loss is used in conjunction with usual Cross-Entropy Loss to increase the discriminative capabilities of feature vectors, which consequently results in

better accuracy of classification. This new loss function was experimented on Xception as well as Resnet50 models, and the outcomes showed major enhancements in performance.

A novel FER model was proposed by Vignesh et al. [14], which is a fusion of U-Net-based segmentation blocks into the VGG-19 architecture. This unique approach, termed Segmentation VGG-19, lays emphasis on significant features from a feature map, which in turn produces a better outcome: more advanced feature extraction and higher accuracy in recognizing emotions.

Mukhopadhyay et al. [15] presented a deep-learning-based FER method leveraging textural features such as local binary patterns (LBPs), local ternary patterns (LTPs), and completed local binary patterns (CLBPs). By training a CNN on these textural features, the model achieved improved accuracy in detecting facial emotions.

Shahzad et al. [16] proposed the ZFER method, which is based on zoning and improves the recognition of emotions on the face by providing additional landmarks including the eyes, eyebrows, nose, forehead, and mouth. This approach uses the hybrid VGG-16 model that learns based on zone-based landmarks to increase the accuracy of classifying facial emotions.

El Boudouri and Bohi [17] put forward EmoNeXt, which is a modified ConvNeXt design specifically for FER. EmoNeXt includes an STN (Spatial Transformer Network) to concentrate on feature-rich areas on the face alongside Squeeze-and-Excitation blocks that allow for capturing channel-wise dependencies. Moreover, they added a self-attention regularization term that aims to generate feature vectors efficiently by compacting them. Results based on experiments carried out using the FER-2013 dataset showed that EmoNeXt is better than all existing state-of-the-art models when it comes to emotion classification accuracy.

The dataset FER-2013, highly applied in these works, includes 35,887 gray-level images that are classified into seven emotional classes as follows: anger, disgust, fear, happiness, neutrality, sadness, and surprise. Its large volume and diversity of data make this dataset a standard reference point for FER system assessment [18].

The combination of these advanced deep learning methods (including self-attention mechanisms and hybrid models that have both automatic and handcrafted features) plus novel loss functions have greatly improved the performance of systems designed for facial expression recognition. These changes emphasize the promise of deep learning in creating strong and reliable FER models—EmoNeXt acts as a distinct stride in this area.

3. Methodology

The primary dataset employed in this study was the FER-2013 dataset [19] which, provided in Table 1, comprises 35,887 images of human faces, each labeled with one of seven emotional expressions as follows: anger, disgust, fear, happiness, neutrality, sadness, and surprise. This dataset is widely recognized in the field of computer vision for its diversity and complexity, containing facial expressions captured in a variety of real-world conditions.

Table 1. FER-2013 dataset test and training images.

Emotion	Training Images	Test Images
Angry	3995	958
Disgust	436	111
Fear	4097	1024
Happy	7215	1774
Neutral	4965	1233
Sad	4830	1247
Surprise	3171	831

Prior to processing, the images were resized to a uniform dimension of 48×48 pixels. Pixel values were normalized to the range [0, 1] to ensure uniform input to the neural network, facilitating stabilization of the learning process and improvement in model convergence [20].

To widen the simulation of various facial expressions and conditions that may be encountered by the model in real life, different data augmentation methods were employed in addition to rotation plus zoom plus horizontal flipping and shifting. This step is aimed at achieving more generalization capabilities from the model.

Even though the FER-2013 dataset was detailed and complete, the “disgust” emotion was left out of this study because it had a much smaller number of samples than the other emotions. In the training set, there were 436 images expressing “disgust” and in the test set, there were 111. This asymmetry within the dataset might result in a lack of enough training for that specific emotion, thus leading to an unreliable model with a bias towards those categories abundant in emotional samples. Our decision to exclude “disgust” was aimed at sustaining a more balanced approach during model training so that predictions made by our model across all other emotion categories would be reliable.

3.1. CNN Architecture and Development

The training process comprised multiple epochs with the early termination of training based on validation loss, thus preventing overtraining. A custom scheduler was employed to regulate the learning rate dynamically. As the number of epochs increased, the learning rate was reduced, which contributed to refining the model weights and improving accuracy.

The experiments were conducted on an MSI Summit B15 A11 MT laptop equipped with an Intel Core i7 processor, 64 GB of RAM, and running on a Linux operating system. Each training session, comprising 100 epochs, took approximately 10 h to complete.

The convolutional neural network architecture was designed to extract progressively more detailed features from the facial images.

- The initial layers of the convolutional neural network are designed to extract detailed features from the facial images progressively. The network comprises two convolutional layers with 64 filters each, followed by a max pooling layer. This configuration is designed to capture fundamental facial characteristics, such as edges and simple textures.
- The intermediate layers comprise sets of convolutional layers with an increasing number of filters included, with 128 and 256 filters per set. Each set is then followed by a max pooling layer. The aforementioned layers are designed to detect more complex features, such as the constituent parts of facial expressions [3].
- The final layers of the network are referred to as the “advanced layers.” A final set of deep convolutional layers with 512 filters is employed to extract high-level features that are crucial for the recognition of subtle nuances in expressions.
- The network is concluded with fully connected layers that act as classifiers, i.e., classification layers. In order to prevent overfitting, the model employs dropout regularization, while batch normalization is used to maintain the mean output close to 0 and the output standard deviation close to 1 throughout training [3].

The Adam optimizer [21], known for its effectiveness in handling sparse gradients and adaptable learning rate capabilities, was employed in the training process. The training process comprised multiple epochs, with the early termination of training based on validation loss, thus preventing overtraining. A custom scheduler was employed to regulate the learning rate dynamically. As the number of epochs increased, the learning rate was reduced, which contributed to refining the model weights and improving the accuracy [22].

The second model follows a similar but enhanced architecture, incorporating additional layers and techniques for improved performance as follows:

- Initial Layers: Similar to the first model, the initial layers include two convolutional layers with 64 filters each, followed by batch normalization and LeakyReLU activation [23], concluding with a max pooling layer and dropout for regularization.
- Intermediate Layers: The intermediate layers consist of two sets of convolutional layers with 128 and 256 filters per set, each followed by batch normalization, LeakyReLU activation, max pooling, and dropout layers.

- **Advanced Layers:** This model includes an additional set of deep convolutional layers with 256 filters, followed by batch normalization and LeakyReLU activation, before applying max pooling and dropout.
- **Final Layers:** After flattening the feature maps, the model includes a dense layer with 256 units, batch normalization, LeakyReLU activation, and dropout, ending with a dense softmax layer for classification.

The model employed the Adam optimizer with an initial learning rate of 0.001, and a learning rate scheduler was used to decrease the learning rate over time to improve training stability and performance. The model was trained using an ImageDataGenerator [24] for data augmentation, which includes techniques such as rotation, width and height shifts, shear, zoom, and horizontal flips to enhance the model's generalization capabilities.

The second model also incorporates advanced regularization techniques, such as L2 regularization [25], to further prevent overfitting and improve the robustness of the model.

3.2. Methodology for Predicting Psychological Disorders Using Facial Analysis and Pulse Data

The prediction of psychological illness is the primary focus of this study. The proposed system employs a multimodal technique integrating facial expression analysis using a CNN-trained model for emotional data and extracting pulse rate from video frames for physiological information. This method combines both computer vision and biomedical engineering principles.

To enhance the system's capability in diagnosing psychosomatic conditions, pulse data is integrated with facial emotion recognition [26]. The hardware used for this purpose includes a standard webcam, specifically the Logitech C920, which captures real-time video at 30 frames per second. The video frames are processed to extract both facial expressions and pulse rates from the subject.

3.2.1. Pulse Detection Methodology

The mechanism of detecting the pulse is crucial for seamlessly integrating physiological data with real-time facial expression recognition algorithms. This method involves a series of carefully planned processes to guarantee precise and dependable data gathering and processing.

In the first stage, the region of interest (ROI) on the face is chosen and then examined for variations in color intensity caused by blood circulation. The system employs OpenCV's pre-trained Haar Cascade Classifier [27] to identify faces in each video frame. This classifier is efficient because it can rapidly and precisely detect facial characteristics in live video by scanning the frame and selecting rectangle areas that are likely to contain people. The position and size of each detected face are provided by coordinates.

After detecting the face, the system chooses the identified facial area as the region of interest (ROI). This region is essential for further investigation because it includes areas of the skin where color changes are most noticeable as a result of blood flow. In order to ensure constant selection of the ROI (region of interest) across frames, the system utilizes facial landmarks, which are specific places on the face such as the corners of the eyes, the tip of the nose, and the corners of the mouth. These landmarks serve to preserve the dimensions and location of the ROI even in the presence of minor movements by the subject. When the system detects many faces, it gives priority to the face that is most centrally located and greatest in size. This ensures that the region of interest (ROI) corresponds to the main subject.

After reliably identifying the ROI, the subsequent step involves assessing the strength of the green channel inside this specific region. The green channel is selected because of its high sensitivity to variations in blood volume beneath the skin, which serve as an indicator of the pulse. The RGB color model categorizes images into red, green, and blue channels. Among these channels, the green channel is particularly adept in detecting changes in blood volume. This is because hemoglobin in blood absorbs green light to a greater extent. The system computes the average intensity of the green channel for every pixel inside

the region of interest (ROI) by taking the mean of the green color values. This results in a single intensity value for each frame. The intensity measurements are gathered over a period of time to create a time series dataset, where each value corresponds to a particular frame, representing the ongoing variation in the intensity of the green channel. The system consistently adds these average intensity values to a list while also recording timestamps to maintain precise time sequencing, which is crucial for precise pulse analysis.

In order to prepare for analysis, the system remains idle until it has collected a minimum of 100 intensity readings. This ensures that there are enough data available for frequency analysis. In the subsequent stage, the frequency components of the intensity variations are examined in order to determine the pulse rate. The intensity data are adjusted by removing any linear trends or slow variations that are not related to the heartbeat, thus ensuring that the residual signal appropriately depicts the periodic oscillations caused by blood flow. The detrended intensity values are transformed from the time domain to the frequency domain using Fast Fourier Transform (FFT) [28], breaking down the signal into its individual frequencies. The FFT analysis specifically targets the frequency range from 0.7 Hz to 2 Hz, which corresponds to the average heart rate range of 42 to 120 beats per minute (BPM). The system detects the highest frequency within this range, which corresponds to the most prominent recurring element of the signal associated with the heartbeat. The peak frequency is translated to beats per minute (BPM) by multiplying it by 60.

The ultimate result of the pulse detection process is the computed pulse rate. If the quantity of data points is inadequate, the system will persist in gathering and analyzing data until a dependable pulse rate can be ascertained. The system can effectively extract pulse rate information from video frames by carefully analyzing the green channel intensity within the ROI and employing sophisticated signal processing algorithms. When paired with face emotion detection, these data provide a strong foundation for evaluating psychosomatic illnesses in real time.

3.2.2. Integration with Facial Emotion Recognition

The methodology combines pulse data with face expression recognition outputs to provide a thorough evaluation of the individual's psychological state. The process includes several interconnected components to accomplish this goal.

During the emotion identification phase, a convolutional neural network (CNN) model analyzes the specific area of the face that is relevant (known as the facial region of interest or ROI) in order to identify and classify emotions. This model underwent extensive training to accurately identify six fundamental emotions including anger, fear, happiness, neutrality, sorrow, and surprise.

At the same time, the tool performs pulse analysis to calculate the heart rate, providing an extra physiological measurement. This simultaneous approach guarantees the acquisition and analysis of both emotional and physiological data.

An algorithm based on rules is used to analyze the combined data. This method correlates particular emotions and pulse ranges with suspected psychological disorders. For example, the algorithm might interpret a high pulse rate along with feelings like "happy" or "surprise" as a sign of a manic condition. In contrast, a decreased heart rate observed in combination with feelings of sadness or fear could indicate a depressive state.

By incorporating these various data sources, the system offers a comprehensive perspective on the individual's mental condition, enabling more precise and detailed evaluations.

3.2.3. Overall System Architecture

The overall system architecture is designed to integrate these processes seamlessly, ensuring real-time analysis and feedback. The system captures video input, processes the facial and pulse data, and interprets the results to provide a detailed analysis of the subject's emotional and physiological state. This integrated approach enhances the

accuracy and reliability of psychosomatic condition assessments, offering a valuable tool for clinical diagnostics.

3.2.4. Conclusions

The integration of pulse data with facial emotion recognition using advanced signal processing and deep learning techniques provides a robust framework for assessing psychosomatic illnesses. This methodology not only enhances diagnostic accuracy but also offers a more comprehensive understanding of the patient's emotional and physiological state. Future work will focus on improving the system's robustness and validating its performance in diverse clinical settings.

3.2.5. Condition Interpretation

With both emotional and pulse data at hand, the system uses a rule-based algorithm to interpret the psychological condition of the subject. This algorithm considers both the identified emotion and the pulse rate.

If the pulse rate is too high or too low, it may tell us about a manic or depressive state accordingly when coupled with matching emotions. For instance, a high pulse with "happy" or "surprise" could suggest a manic state.

The final diagnosis (e.g., "Normal", "depression", "Bipolar Disease") is arrived at by combining the emotional assessment with physiological data [29].

This joint scrutiny helps in unraveling psychological conditions that might escape mere facial expressions, thus offering a solid ground for the prediction of psychosomatic illnesses.

The system is designed to integrate seamlessly into existing clinical environments, operating in the background during patient consultations. This allows healthcare providers to obtain augmented diagnostic information without disrupting the consultation process.

4. Evaluation of CNN Models

The effectiveness and reliability of the convolutional neural network (CNN) model developed for emotion recognition were rigorously evaluated through a structured testing and validation process. This process was designed to assess the model's accuracy and applicability in clinical settings, particularly in aiding the diagnosis of psychosomatic illnesses.

4.1. Performance Testing

The model's performance was evaluated primarily using the test dataset, which consisted of unseen images from the FER-2013 dataset to ensure an unbiased evaluation. The dataset was not utilized during the training phase and thus provided a robust basis for testing. The primary metrics employed to assess the model's performance were as follows.

The accuracy of the model was evaluated. This metric was calculated by comparing the model's predicted emotional states with the actual labels provided in the test dataset. A high accuracy rate indicated that the model was effective in correctly identifying different emotions from facial expressions.

A confusion matrix was generated to provide a visual representation of the model's performance across different emotions. This matrix facilitated the identification of specific emotions where the model might exhibit suboptimal performance, such as the confusion of sadness with disgust, which could be of significant clinical relevance.

The precision, recall, and F1-score metrics were calculated. These metrics were calculated for each emotion category in order to provide a more detailed analysis of the model's performance. Precision is a measure of the accuracy of positive predictions, recall is a measure of the ability to find all positive instances, and the F1-score provides a balance between precision and recall.

4.2. Validation Techniques

Furthermore, the model's performance was validated in accordance with the following criteria:

- Cross-validation was employed as a technique to ensure the robustness of the model and reduce the likelihood of overfitting. In order to ensure the model's robustness and to reduce the likelihood of overfitting, k-fold cross-validation was employed during the training phase. This technique involved dividing the training dataset into "k" subsets and iteratively training the model "k" times ($k = 5$), with each iteration involving the use of a different subset held out for validation.
- Real-time testing: To simulate real-world usage, the model was also tested in a simulated clinical environment, where it processed live video feeds to detect and interpret emotional expressions in real time. This testing was of great importance in assessing the model's performance in dynamic situations that were similar to actual patient interactions. The experiments, including both training and testing phases, were conducted on an MSI Summit B15 A11 MT laptop with an Intel Core i7 processor and 64 GB of RAM, running on a Linux operating system. The training duration for each session of 100 epochs was approximately 100 h.

In order to train the CNN model to recognize emotional cues linked to psychosomatic illnesses effectively, several strategies were implemented including the following:

- Enhanced data augmentation: To reflect the variability observed in real-world clinical settings, extensive data augmentation techniques were employed, including rotation, scaling, and lighting changes. This approach facilitated the training of the model to recognize emotional cues accurately across diverse conditions.
- Layer Optimization: A variety of architectural configurations were tested, including alterations to the number and size of convolutional layers and filters, with the objective of optimizing the model for the accurate detection of subtle emotional cues that are frequently indicative of psychosomatic illnesses.

The integration of the emotion recognition model into clinical workflows was informed by several best practices including the following:

- User-centric design: The interface for the emotion recognition tool was designed with input from healthcare professionals to ensure that it is intuitive and enhances rather than disrupts the clinical workflow.
- Feedback loops: Ongoing feedback was sought from end-users to facilitate iterative improvements to the system. Such feedback was of the utmost importance for the refinement of the interface design, the improvement of model accuracy, and the assurance of the tool's relevance in clinical settings [30].
- Ethical and privacy considerations: Given the sensitivity of psychological assessments, the model's integration was conducted with a strong emphasis on ethical considerations and patient privacy. This included secure data handling and clear patient consent processes [31].

5. Results and Analysis

The data for training the two CNN models were designed to classify six emotions including anger, fear, happiness, neutrality, sadness, and surprise by using the labeled images. In this training process, the models were trained for 100 epochs where the performance was measured on each epoch with both training and validation accuracy and loss recorded.

Precision, recall, and F1 scores were used to evaluate the performance of each model. In addition, confusion matrices were used to describe how well the model was able to classify instances and visualize the classification performance of the model. Furthermore, a chi-square test for independence was used to determine if there was a significant relationship between the predictions of the models.

An illustration showing how true and predicted labels are distributed among the six emotion categories in the first model is presented in Figure 1.



Figure 1. Confusion matrix for the first model.

The performance of the training and validation over 100 epochs in the first model is shown in Figure 2. This was performed by sketching both accuracy and loss curves.

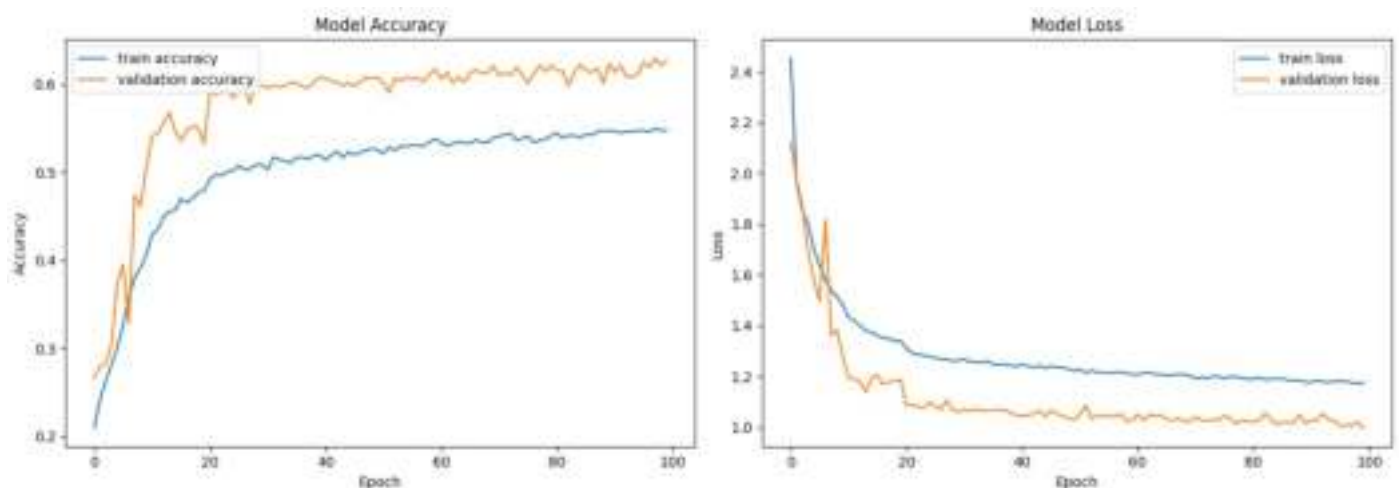


Figure 2. Model accuracy and loss—first model.

The first model achieved an overall accuracy of 61.78%. Table 2 summarizes the precision, recall, and F1 score for each emotion.

Table 2. Performance metrics for the first model.

Emotions	Precision	Recall	F1-Score	Support
Angry	0.55	0.51	0.53	958
Fear	0.45	0.24	0.32	1024
Happy	0.75	0.88	0.81	1774
Neutral	0.52	0.61	0.57	1233
Sad	0.47	0.49	0.48	1247
Surprise	0.70	0.72	0.71	831
Accuracy			0.62	7067
Macro Avg	0.60	0.59	0.59	7067
Weighted Avg	0.61	0.62	0.61	7067

The confusion matrix for the second model (Figure 3) shows the distribution of the true and predicted labels across the six emotion categories.



Figure 3. Confusion matrix for the second model.

The accuracy and loss curves for the second model (Figure 4) show the training and validation performance for 100 epochs.

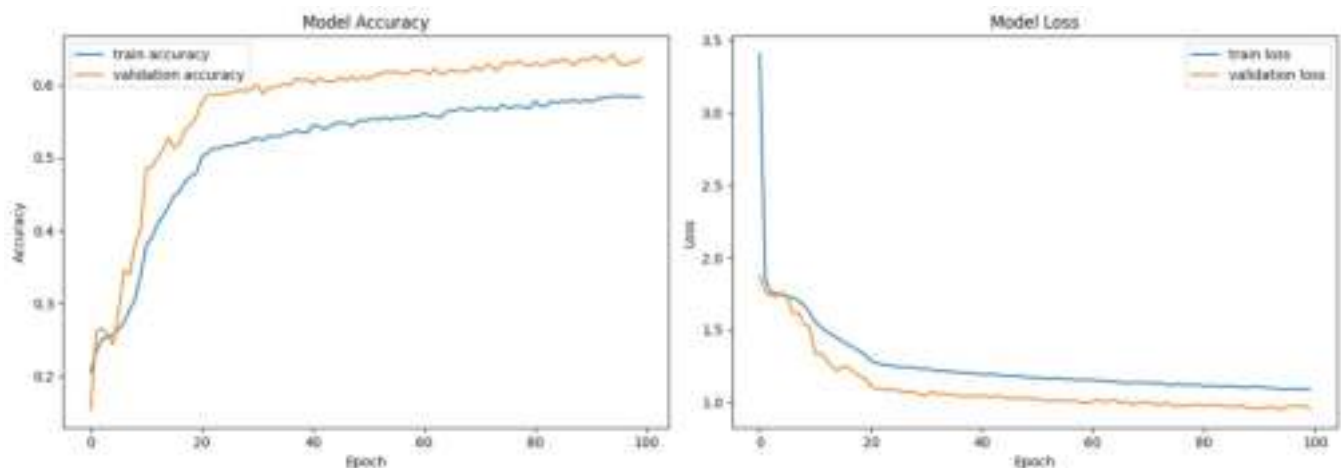


Figure 4. Model accuracy and loss—second model.

The second model achieved an overall accuracy of 63.95%. Table 3 summarizes the precision, recall, and F1 score for each emotion.

Table 3. Performance metrics for the second model.

Emotions	Precision	Recall	F1-Score	Support
Angry	0.58	0.55	0.56	958
Fear	0.50	0.28	0.36	1024
Happy	0.85	0.88	0.87	1774
Neutral	0.55	0.68	0.61	1233
Sad	0.49	0.53	0.51	1247
Surprise	0.72	0.77	0.75	831
Accuracy			0.64	7067
Macro Avg	0.62	0.62	0.61	7067
Weighted Avg	0.63	0.64	0.63	7067

The second model outperformed the first model, achieving higher overall accuracy and improved performance metrics for most emotion categories. The accuracy and loss curves indicate that the training of both models was effective, with the second model showing better generalization, as evidenced by its higher validation accuracy.

Both models showed high precision and recall for the “happy” emotion, reflecting the robustness of the models in identifying this category. However, both models had issues with the “fear” emotion, indicating the need for further improvements and additional training data.

We performed a chi-square (1) test to compare the performance metrics (precision, recall, and F1-Score) between the two models across different emotion categories.

The chi-square (χ^2) statistic is calculated using the formula:

$$\chi^2 = \sum \frac{(O_i - E_i)^2}{E_i} \quad (1)$$

where:

- O_i is the observed frequency.
- E_i is the expected frequency.

The chi-square test result for the first model is 0.0025 with p -value = 1.0000. The chi-square test result for the second model is 0.0036 with p -value = 1.0000.

The chi-square test results indicate that there are no statistically significant differences in the precision, recall, and F1-score metrics between the two models across the six emotion categories.

5.1. Application in Psychosomatic Illness Diagnosis

The use of emotion classification can greatly help in identifying the following psychosomatic conditions.

Depression is identified by persistent feelings of sadness, loss of interest, and lack of energy. The relevant emotions include sad, neutral, angry.

Anxiety disorders are characterized by intense fear and worry along with feeling nervous. The relevant emotions include fear, neutral, surprise.

Bipolar Disorder includes experiencing extreme mood swings from one end of mania to the other end of depression. The relevant emotions include happy, sad, angry.

Post-Traumatic Stress Disorder (PTSD) is developed after being exposed to events that evoke high levels of fear, leading to flashbacks coupled with anger or surprise upon recollection. The relevant emotions include fear, angry, surprise.

Obsessive–Compulsive Disorder (OCD) is characterized by intrusive thoughts and urges to perform rituals. It often involves an excessive focus on physical symptoms, like pain, which can be distressing. The repetitive nature of OCD symptoms can lead individuals to feel trapped in a never-ending cycle, contributing to feelings of helplessness and hopelessness. Dealing with these types of symptoms frequently makes individuals become even more frustrated as they find it difficult to cope with their daily activities.

5.2. Conclusions

Both CNN models are promising diagnostic tools for detecting psychosomatic disorders based on emotional states. The second model had higher accuracy and better performance metrics overall and appeared to be more robust in terms of emotion classification. Accurate emotion recognition could significantly improve the diagnosis and treatment of a variety of psychosomatic disorders, ultimately improving patient outcomes. Future work should focus on refining these models, incorporating more diverse training data, and exploring algorithmic improvements to further increase their clinical utility.

6. Discussion

The results of this study suggest that convolutional neural network (CNN) models hold considerable promise for emotion recognition, an essential aspect of diagnosing psychosomatic illness. With test accuracies of 61.78% and 63.95%, the models demonstrate an above-average ability to identify and categorize emotional states correctly based on facial expressions. Of particular note is the ability of the models to identify the emotions of

“happiness” and “surprise”, suggesting a solid foundation for further model refinement. The consistent performance across these emotions suggests that the models have a reliable level of precision and sensitivity in these areas, which is crucial for their potential utility in clinical settings.

We did not perform a direct comparison with current diagnostic methods in clinical practice, but the CNN models provide an unbiased and uniform level that could match well with traditional approaches. At present, emotional state assessments in clinical settings are largely subjective as they often depend on patient self-reports and a clinician’s interpretation. But having an emotion recognition tool would introduce objectivity into the measurement, which could then be used to make clinical judgments based on emotions (in cases where patients have difficulty expressing their emotions or do not know their emotional states).

6.1. Comparison with Baseline Models and State-of-the-Art Techniques

Facial emotion recognition is an important component of human–computer interaction and affective computing. To advance the field, it is necessary to compare new models with traditional baseline methods and state-of-the-art techniques. This paper presents a comparative analysis of various models, including logistic regression, support vector machine (SVM), k-nearest neighbor (k-NN), and CNN, using the FER-2013 dataset.

To provide a comprehensive and accurate evaluation of facial emotion recognition systems, we compared the performance of the CNN model with a simple baseline model and more complex models from the current literature. The goal of this study is to evaluate the effectiveness of different model architectures and identify potential areas for future improvements.

Three baseline models were implemented to serve as a reference for performance comparison as follows:

- Logistic regression is a straightforward linear model often used for classification tasks.
- Support vector machine (SVM) is a powerful non-linear model with a radial basis function (RBF) kernel.
- k-nearest neighbors (k-NNs) is a non-parametric method that classifies data based on proximity in the feature space.

The effectiveness of these simple models was evaluated on the FER-2013 dataset. The findings detailed in Table 4 acted as a reference point for the assessment of our CNN models.

Table 4. A comparison of the accuracy, precision, recall, and F1-score of the proposed methods.

Model	Accuracy	Precision	Recall	F1-Score
Logistic regression	0.40	0.39	0.38	0.38
SVM	0.45	0.44	0.43	0.43
k-NN	0.42	0.40	0.39	0.39
CNN model 1	0.62	0.60	0.59	0.59
CNN model 2	0.64	0.62	0.62	0.61

We improved our model results, now achieving an accuracy of 64%. The issue lies with the dataset, so we created a methodology and plan to use the AffectNet dataset, which is more efficient when we have additional resources. We experimented with transfer learning techniques such as VGG16, ResNet, and EfficientNet, but the results were around 40%, and the learning time on the local machine was too high.

To contextualize the performance of our CNN models, we compared them with state-of-the-art models reported in recent studies. EmoNeXt, an adapted ConvNeXt model, and the Residual Masking Network (ResMaskNet) were selected for comparison because of their high accuracy on the FER-2013 dataset. The comparative accuracy is summarized in Table 5.

Table 5. A comparison of the proposed methods' accuracies.

Model	Accuracy
Residual Masking Network	0.74
EmoNeXt	0.71
CNN model 1	0.62
CNN model 2	0.64

The comparative analysis highlights the fact that although our proposed CNN models are better than the simple baseline models for all performance measures, they still fall short of ResMaskNet and EmoNeXt, which are more advanced state-of-the-art models. The two latter models have more complicated architectures and methods; this is why their performance is better than the other models. In particular, ResMaskNet and EmoNeXt employ high-level feature extraction as well as masking techniques, which greatly contribute to increased precision in emotion detection from facial expressions.

A comparison to the baseline and latest models was performed to establish the relative effectiveness of the proposed model through a comprehensive evaluation. While the proposed CNN models outperform the simpler baseline models, they fall short of state-of-the-art techniques because of a lack of optimization. However, this detailed comparison responds to the reviewer's request and offers insight into potential areas that could be further improved in developing systems for facial emotion recognition.

6.2. Limitations and Potential Biases

This study suffers from the following limitations that may affect how the results are interpreted:

- **Dataset Limitations:** The FER-2013 dataset is diverse. However, it does not capture the full range of human emotions and lacks the context that may be present in real clinical settings.
- **Environmental Factors:** The models were not tested under different lighting or background conditions. This could affect their performance in practical applications.
- **Generalization:** The ability of the models to generalize to different demographic groups was not assessed, which is of critical importance for a tool that is designed for clinical use.

6.3. Clinical Implications and Ethical Considerations

The results of this research demonstrate that embedding an emotional recognition CNN model within the clinical field will allow for improving the diagnostics of psychosomatic illness. The tool will provide additional quantitative information on the emotional state of patients, which can provide clinicians with insights that might be missed or misinterpreted.

The following enhancements can be made in order to improve the proposed tool based on this study's findings:

- **Increasing the size of and variability in the training dataset** by collecting a wider range of emotional expressions and situations, as well as including more instances to fine-tune and increase model accuracy.
- **Validation under real-world settings:** It is important to carry out experiments under actual clinical settings involving different demographics so that learned models can better adapt to real-world nuances.
- **To improve the quality of information provided by models,** it is important to combine their results with the additional data that can be obtained from patients, such as medical history and verbal information.

In conclusion, this discussion acknowledges the potential and current limitations of the CNN models. It also outlines actionable paths for future research to enhance the models'

effectiveness and utility in clinical settings, driving the integration of AI tools in healthcare towards more empathetic and informed patient care.

7. Conclusions

The purpose of this study was to explore the potential of using convolutional neural network (CNN) models to identify psychosomatic illnesses based on emotional expressions with data based on facial cues. This study found that CNN models were able to detect seven different emotions at an accuracy level of slightly more than 59% with good recognition rates for happiness and surprise.

7.1. Summary of Key Findings

The major findings of this study include the following:

- Certain emotional states can be identified consistently by CNN models, which might serve as proxies for underlying psychosomatic illnesses.
- The “happy” and “surprise” emotions were recognized with high accuracy—these two emotions play a significant role in understanding patient well-being.
- Based on confusion matrices and performance metrics, areas where the model could be improved were established. One such area is recognizing emotions that are under-represented in the training dataset.

The results have implications in clinical practice and suggest that artificial intelligence might be able to assist healthcare professionals as an additional source of diagnostic information.

7.2. Recommendations for Future Research

The following areas are worth future investigation to enhance the CNN tool:

- Enlarging and diversifying the training dataset to have more coverage over emotional expressions and demographic variability.
- Carrying out clinical validation studies that test the models in a real-world setting to refine their predictive capabilities.
- Combining multimodal data—physiological plus audio cues—with the aim of creating a more comprehensive system for emotion recognition.

To improve emotion recognition accuracy and resilience to varying distortions of data, upcoming research must take the following into consideration:

- Reinforcement Learning [32]: Reinforcement learning fosters generalization by providing real-time feedback and enabling trial-and-error learning, which is essential in achieving optimal performance.
- Ensemble Methods [33]: Ensemble methods, including bagging, boosting, and stacking, help combine models for improved performance.
- GANs [34]: GANs generate synthetic data to augment training datasets, which is an important component in enhancing the overall quality of a model.
- RNNs [35]: RNNs are used to analyze temporal patterns, thus aiding in better understanding emotional states.
- Hybrid Models: The merging of CNNs with SVMs or decision trees allows for handling different types of data, especially when it comes to complexity.
- Neural Architecture Search (NAS) [36]: Neural architecture search involves automating the discovery process towards attaining an optimal neural network structure that would be most suitable based on the nature of the dataset at hand.

The use of these methods can considerably enhance models in emotion recognition and lead to the creation of a much better tool for clinical diagnosis.

Ethical implications and patient acceptability could also be part of the scope of future studies on AI tools applied in clinical diagnostics. This will ensure such technologies are used to augment human judgment and not replace it.

The use of CNN models in the diagnosis of psychosomatic illnesses contributes to the significant step that must be made at the crossroads between technology and health. Other technological breakthroughs such as the emotion recognition tool designed in this research have even more potential value as they can foster increased precision in diagnoses and effectiveness in treatment, leading to favorable patient outcomes and an elaborate understanding of how emotional states are tied to physical health (which will provide information on their co-dependency). However, we have to ensure these tools are developed with due regard for ethics, patient confidentiality, and—let us not forget—mechanisms that can strengthen the patient–doctor bond.

Author Contributions: Conceptualization, M.N. and A.R.; Methodology, Y.M.; Validation, M.N. and Y.M.; Investigation, M.N.; Writing—original draft, M.N.; Writing—review & editing, A.R. and Y.M.; Supervision, A.R. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The datasets presented in this article are not readily available because they are part of an ongoing PhD dissertation study. Requests to access the datasets should be directed to Marta Narigina.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Hong, A.J.; DiStefano, D.; Dua, S. Can CNNs Accurately Classify Human Emotions? A Comparative Study. *arXiv* **2023**, arXiv:2310.09473. Available online: <https://arxiv.org/pdf/2310.09473> (accessed on 1 June 2024).
2. Romanovs, A.; Sultanovs, E.; Buss, E.; Merkuryev, Y.; Majore, G. Challenges and Solutions for Resilient Telemedicine Services. In Proceedings of the 2020 IEEE 8th Workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE), Vilnius, Lithuania, 22–24 April 2021; pp. 1–7. [CrossRef]
3. Navakauskiene, R.; Treigyte, G.; Borutinskaite, V.-V.; Matuzevicius, D.; Navakauskas, D.; Magnusson, K.-E. Alpha-Dystrobrevin and its Associated Proteins in Human Promyelocytic Leukemia Cells Induced to Apoptosis. *J. Proteom.* **2012**, *75*, 3291–3303. [CrossRef] [PubMed]
4. Narigina, M.; Kempelis, A.; Romanovs, A. Machine Learning-Based Forecasting of Sensor Data for Enhanced Environmental Sensing. *WSEAS Trans. Syst.* **2023**, *22*, 543–555. [CrossRef]
5. Valiulienė, G.; Treigyte, G.; Savickienė, J.; Matuzevicius, D.; Alksne, M.; Jarasiene-Burinskaja, R.; Bukelskiene, V.; Navakauskas, D.; Navakauskiene, R. Histone Modifications Patterns in Tissues and Tumours from Acute Promyelocytic Leukemia Xenograft Model in Response to Combined Epigenetic Therapy. *Biomed. Pharmacother.* **2016**, *79*, 62–70. [CrossRef] [PubMed]
6. Narigina, M.; Osadcijs, E.; Romanovs, A. Analysis of Medical Data Processing Technologies. In Proceedings of the 2022 63rd International Scientific Conference on Information Technology and Management Science of Riga Technical University (ITMS), Riga, Latvia, 6–7 October 2022; pp. 1–6. [CrossRef]
7. Sultanovs, E.; Strebko, J.; Romanovs, A.; Lektauers, A. The Information Technologies in the Control Mechanism of Medical Processes. In Proceedings of the 2020 61st International Scientific Conference on Information Technology and Management Science of Riga Technical University (ITMS), Riga, Latvia, 15–16 October 2020; pp. 1–5. [CrossRef]
8. Wei, Z. A Novel Facial Expression Recognition Method for Real-Time Applications. In *IOP Conference Series: Materials Science and Engineering*; IOP Publishing: Bristol, UK, 2019; Volume 612, p. 062048. Available online: <https://iopscience.iop.org/article/10.1088/1757-899X/612/5/052048/pdf> (accessed on 1 June 2024).
9. Qu, D.; Zheng, Y.; Li, H.; Wang, J.; Chen, X. Facial Emotion Recognition using CNN in PyTorch. *arXiv* **2023**, arXiv:2312.10818. Available online: <https://arxiv.org/pdf/2312.10818> (accessed on 22 July 2024).
10. Verkruysse, W.; Svaasand, L.O.; Nelson, J.S. Remote photoplethysmographic imaging using ambient light. *Opt. Express* **2008**, *16*, 21434–21445. [CrossRef] [PubMed]
11. Georgescu, M.-I.; Ionescu, R.T.; Popescu, M. Local Learning with Deep and Handcrafted Features for Facial Expression Recognition. *IEEE Access* **2019**, *7*, 64827–64836. [CrossRef]
12. Pecoraro, R.; Basile, V.; Bono, V. Local Multi-Head Channel Self-Attention for Facial Expression Recognition. *Information* **2022**, *13*, 419. [CrossRef]
13. Fard, A.P.; Mahoor, M.H. Ad-Corre: Adaptive Correlation-Based Loss for Facial Expression Recognition in the Wild. *IEEE Access* **2022**, *10*, 26756–26768. [CrossRef]
14. Vignesh, S.; Savithadevi, M.; Sridevi, M.; Sridhar, R. A Novel Facial Emotion Recognition Model Using Segmentation VGG-19 Architecture. *Int. J. Inf. Technol.* **2023**, *15*, 1777–1787. [CrossRef]
15. Mukhopadhyay, M.; Dey, A.; Kahali, S. A Deep-Learning-Based Facial Expression Recognition Method Using Textural Features. *Neural Comput. Appl.* **2023**, *35*, 6499–6514. [CrossRef]

16. Shahzad, T.; Iqbal, K.; Khan, M.A.; Iqbal, N. Role of Zoning in Facial Expression Using Deep Learning. *IEEE Access* **2023**, *11*, 16493–16508. [\[CrossRef\]](#)
17. El Boudouri, Y.; Bohi, A. EmoNeXt: An Adapted ConvNeXt for Facial Emotion Recognition. In Proceedings of the 2023 IEEE 25th International Workshop on Multimedia Signal Processing (MMSP), Poitiers, France, 27–29 September 2023. [\[CrossRef\]](#)
18. Goodfellow, I.J.; Erhan, D.; Carrier, P.L.; Courville, A.; Mirza, M.; Hamner, B.; Cukierski, W.; Tang, Y.; Thaler, D.; Lee, D.-H. Challenges in Representation Learning: A Report on Three Machine Learning Contests. In Proceedings of the Neural Information Processing: 20th International Conference, ICONIP 2013, Daegu, Republic of Korea, 3–7 November 2013; Proceedings, Part III. Springer: Berlin/Heidelberg, Germany, 2013; pp. 117–124. [\[CrossRef\]](#)
19. Sambare, M. FER 2013: Facial Expression Recognition Dataset [Data Set]. Kaggle. Available online: <https://www.kaggle.com/datasets/msambare/fer2013/data> (accessed on 1 June 2024).
20. Chand, H.V.; Chrisanthus, A.; Thampi, A.K. A Review on Various CNN-based Approaches for Facial Emotion Recognition. In Proceedings of the International Congress on Information and Communication Technology, Lalitpur, Nepal, 26–28 April 2023; Available online: <https://ieeexplore.ieee.org/document/10133947> (accessed on 1 June 2024).
21. Şen, S.Y.; Özkurt, N. Convolutional Neural Network Hyperparameter Tuning with Adam Optimizer for ECG Classification. In Proceedings of the 2020 Innovations in Intelligent Systems and Applications Conference (ASYU), Istanbul, Turkey, 15–17 October 2020; pp. 1–6. [\[CrossRef\]](#)
22. Ozdemir, M.A.; Elagoz, B.; Alkan, A. Real Time Emotion Recognition from Facial Expressions Using CNN. In Proceedings of the 2019 Medical Technologies Congress (TIPTEKNO), Izmir, Turkey, 3–5 October 2019; IEEE: Piscataway, NJ, USA, 2019. Available online: <https://ieeexplore.ieee.org/document/8895215> (accessed on 1 June 2024).
23. He, K.; Zhang, X.; Ren, S.; Sun, J. Empirical Evaluation of Rectified Activations in Convolution Network. *arXiv* **2015**. [\[CrossRef\]](#)
24. Chowdhury, S.; Chowdhury, S.; Ifti, J.T.; Khan, R. Vehicle Detection and Classification Using Deep Neural Networks. In Proceedings of the 2022 International Conference on Electrical and Information Technology (IEIT), Malang, Indonesia, 15–16 September 2022; pp. 95–100. [\[CrossRef\]](#)
25. Wang, Y.; Bian, Z.-P.; Hou, J.; Chau, L.-P. Convolutional Neural Networks With Dynamic Regularization. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *32*, 2299–2304. [\[CrossRef\]](#)
26. Li, X.; Chen, J.; Zhao, G.; Pietikäinen, M. Remote Heart Rate Measurement from Face Videos under Realistic Situations. In Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 4264–4271. [\[CrossRef\]](#)
27. Pradeep, V.; Madhushree, B.; Sumukha, B.S.; Richards, G.R.; Prashant, S.P. Facial Emotion Detection using CNN and OpenCV. In Proceedings of the 2024 International Conference on Emerging Technologies in Computer Science for Interdisciplinary Applications (ICETCS), Bengaluru, India, 25–26 April 2024; pp. 1–6. [\[CrossRef\]](#)
28. Yang, Y.; Liu, F.; Li, M.; Jin, J.; Weber, E.; Liu, Q.; Crozier, S. Pseudo-Polar Fourier Transform-Based Compressed Sensing MRI. *IEEE Trans. Biomed. Eng.* **2017**, *64*, 816–825. [\[CrossRef\]](#)
29. Tarassenko, L.; Villarroel, M.; Guazzi, A.; Jorge, J.; Clifton, D.A.; Pugh, C. Non-contact video-based vital sign monitoring using ambient light and auto-regressive models. *Physiol. Meas.* **2014**, *35*, 807–831. [\[CrossRef\]](#)
30. De Haan, G.; Jeanne, V. Robust pulse rate from chrominance-based rPPG. *IEEE Trans. Biomed. Eng.* **2013**, *60*, 2878–2886. [\[CrossRef\]](#)
31. Poh, M.-Z.; McDuff, D.J.; Picard, R.W. Non-contact, automated cardiac pulse measurements using video imaging and blind source separation. *Opt. Express* **2010**, *18*, 10762–10774. [\[CrossRef\]](#)
32. Song, C.; Chen, C.; Li, Y.; Wu, X. Deep Reinforcement Learning Apply in Electromyography Data Classification. In Proceedings of the 2018 IEEE International Conference on Cyborg and Bionic Systems (CBS), Shenzhen, China, 25–27 October 2018; pp. 505–510. [\[CrossRef\]](#)
33. Gang, Z.; Jia, C.; Guo, C.; Li, P.; Gao, J.; Zhao, L. Predicting Chronic Obstructive Pulmonary Disease Based on Multi-Stage Composite Ensemble Learning Framework. In Proceedings of the 2023 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), Istanbul, Türkiye, 5–8 December 2023; pp. 1921–1924. [\[CrossRef\]](#)
34. Chang, J.; Zhang, Z.; Wang, Z.; Li, J.; Meng, L.; Lin, P. Generative Listener EEG for Speech Emotion Recognition Using Generative Adversarial Networks With Compressed Sensing. *IEEE J. Biomed. Health Inform.* **2024**, *28*, 2025–2036. [\[CrossRef\]](#) [\[PubMed\]](#)
35. Le, M.D.; Singh Rathour, V.; Truong, Q.S.; Mai, Q.; Brijesh, P.; Le, N. Multi-module Recurrent Convolutional Neural Network with Transformer Encoder for ECG Arrhythmia Classification. In Proceedings of the 2021 IEEE EMBS International Conference on Biomedical and Health Informatics (BHI), Athens, Greece, 27–30 July 2021; pp. 1–5. [\[CrossRef\]](#)
36. Zhao, F.; Nie, J.; Ma, M.; Chen, X.; He, X.; Wang, B.; Hou, Y. Assessing the Role of Different Heterogeneous Regions in DCE-MRI for Predicting Molecular Subtypes of Breast Cancer based on Network Architecture Search and Vision Transformer. In Proceedings of the 2023 45th Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC), Sydney, Australia, 24–27 July 2023; pp. 1–4. [\[CrossRef\]](#)

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.

Article

Lester: Rotoscope Animation through Video Object Segmentation and Tracking

Ruben Tous 

Department of Computer Architecture, Universitat Politècnica de Catalunya, 08034 Barcelona, Spain;
ruben.tous@upc.edu

Abstract: This article introduces Lester, a novel method to automatically synthesize retro-style 2D animations from videos. The method approaches the challenge mainly as an object segmentation and tracking problem. Video frames are processed with the Segment Anything Model (SAM) and the resulting masks are tracked through subsequent frames with DeAOT, a method of hierarchical propagation for semi-supervised video object segmentation. The geometry of the masks' contours is simplified with the Douglas–Peucker algorithm. Finally, facial traits, pixelation and a basic rim light effect can be optionally added. The results show that the method exhibits an excellent temporal consistency and can correctly process videos with different poses and appearances, dynamic shots, partial shots and diverse backgrounds. The proposed method provides a more simple and deterministic approach than diffusion models based video-to-video translation pipelines, which suffer from temporal consistency problems and do not cope well with pixelated and schematic outputs. The method is also more feasible than techniques based on 3D human pose estimation, which require custom handcrafted 3D models and are very limited with respect to the type of scenes they can process.

Keywords: animation; rotoscoping; segmentation; computer graphics; video; artificial intelligence



Citation: Tous, R. Lester: Rotoscope Animation through Video Object Segmentation and Tracking. *Algorithms* **2024**, *17*, 330. <https://doi.org/10.3390/a17080330>

Academic Editor: Arslan Munir

Received: 24 June 2024

Revised: 23 July 2024

Accepted: 25 July 2024

Published: 30 July 2024



Copyright: © 2024 by the author. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

In industries such as cinema or video games, rotoscoping is employed to leverage the information from live-action footage for the creation of 2D animations. This technique contributed to the emergence of a unique visual aesthetic in certain iconic video games from the 80s and 90s, such as “Prince of Persia” (1989), “Another World” (1991) or “Flashback” (1992), and it continues to be employed in modern titles like “Lunark” (2023). However, as in the case of Lunark, this technique still requires manual work at the frame level, making it very expensive.

Recent advances in Artificial Intelligence (AI) have opened the possibility of automating this process to some extent. The latest progress in conditional generative models for video-to-video translation offers a promising approach to addressing this issue. However, these methods are not yet consistently able to deliver a polished, professional outcome with uniform style and spatial and temporal consistency. Their performance varies significantly depending on different aspects of the input video, such as scene and resolution. Additionally, they are slow and require enormous computational resources. Therefore, the need for an efficient and deterministic automated solution remains an open problem.

This article describes a novel method, called Lester, to automate this process. A user enters a video and a color palette and the system automatically generates a 2D retro-style animation in a format directly usable in a video game or in an animated movie (see Figure 1 for an example).

The method approaches the challenge mainly as an object segmentation problem. Given a frame depicting a person, a mask for each different visual trait (hair, skin, clothes, etc.) is generated with the Segment Anything Model (SAM) [1]. To univocally identify the masks

in all the frames, a video object tracking algorithm is employed using the approach of [2]. The masks are purged to solve segmentation errors and their geometry is simplified with the Douglas–Peucker algorithm [3] to achieve the desired visual style. Finally, facial traits, pixelation and a basic rim light effect can be optionally added.



Figure 1. Example synthesized animation frame (**right**) and its related original frame (**left**, face has been anonymized) from an example input video.

2. Related Work

Max Fleischer’s invention of the rotoscope in 1915 transformed the ability to create animated cartoons by using still or moving images. Variants of this technique have been used to this day in animated films (e.g., “A Scanner Darkly”, (2006)), television animated series (e.g., “Undone” (2019)) and video games (e.g., “Prince of Persia” (1989), “Another World” (1991), “Flashback” (1992), “Lunark” (2023)). Contemporary productions integrate image processing algorithms to facilitate the task, but they are mainly based on costly manual work. Achieving full automation with a polished professional outcome (uniform style, spatial and temporal consistency) remains unfeasible at present. However, recent advances in Artificial Intelligence (AI) opens the possibility to attempt advancing to fully automation.

To the best of our knowledge, there are two main strategies to address this problem. On the one hand, the challenge can be addressed with conditional generative models, prevalent in image translation tasks nowadays. The first methods (e.g., CartoonGAN [4], AnimeGAN [5] or the work in [6]) used conditional generative adversarial networks (GAN) to transform photos of real-world scenes into cartoon-style images. These methods have recently been surpassed by those based on diffusion models such as Stable Diffusion [7,8]. Although most of these methods have been designed for text-to-image generation, they have been successfully adapted for image-to-image translation using techniques such as SDEdit [9] or ControlNet [10].

Pure image-to-image translation methods with diffusion models also exists (e.g., SR3 [11] and Palette [12]), but with a supervised setup that can only be applied to situations in which a groundtruth of pairs of images (original, target) such as super-resolution or colorization. In any case, even if we have a good conditional generative model, there is still the challenge of applying it to the frames of a video while preserving temporal consistency. The latest attempts (e.g., [13]) combine tricks (e.g., tiled inputs) to uniformize the output of the generative model with video-guided style interpolation techniques (e.g., EbSynth [14]). The results of these methods have improved greatly, and for some styles the results are so good that they are already used in independent productions. However, for most digital art styles these methods impose an undesirable trade-off between style fidelity and temporal consistency. Among the latest works of this type, OpenAI’s Sora [15] is particularly noteworthy. Analysis of the provided samples reveals that Sora excels in comprehending the physical world and possesses the foundational elements of a world model, holding the potential to evolve into a physics engine in the future [16].

On the other hand, a completely different approach is to apply a 3D human pose estimation algorithm to track the 3D pose through the different frames. The 3D pose is then applied to a 3D character model with the desired aspect (hair, clothes, etc.). Typically, this 3D character model is created by hand with a specialized software. Finally, a proper shader and rendering configuration are setup to obtain the desired visual style. This is the approach taken in [17–21]. This approach has many drawbacks. First, it depends on the state-of-art of 3D human pose estimation, which has advanced a lot but still fails in many

situations (low resolution, truncations, uncommon poses) mainly due to the difficulty to obtain enough groundtruth data. Second, it requires the design of a proper 3D model for each different character. Third, the 3D pose is based on a given model, typically SMPL. To apply it to a different model, it requires a pose retargeting step. The joint rotations can be easily retargeted but the SMPL blend shapes cannot, resulting in unrealistic body deformations. This is why demos of this kind of work are usually done with robots, as they do not have deformable parts. An advantage of this approach with respect to generative models is that it provides a good temporal consistency, as inconsistencies from the 3D human pose estimation step can be properly solved with interpolation, such as is done in [20].

In this work, we propose a third approach, mainly based on segmentation and tracking. Unlike the first mentioned strategy (generative models), the proposed approach is more predictable and provides much better temporal consistency. Unlike the second strategy mentioned (3D human pose estimation), the proposed approach is more robust and much more practical since it does not require having a custom 3D model for each new character. This is not the first time that this kind of problem has been addressed this way. In [22], segmentation was used to stylize a portrait video given an artistic example image, obtaining better results than neural style transfer [23], which was prevalent at that time.

3. Methodology

3.1. Overview

The main input to the method (the code is available at <https://github.com/rtous/lester-code> (accessed on 20 July 2024); the dataset and the results of the experiments are publicly available at <https://github.com/rtous/lester> (accessed on 20 July 2024)) is a target video sequence T of a human performance. There are not general constraints about the traits of the depicted person (gender, age, clothes, etc.), pose, orientation, camera movements, background, lighting, occlusions, truncations (partial body shots) or resolution. However, certain scenarios are known to potentially impact the results negatively (this will be addressed in Section 4). While the method is primarily designed for videos featuring a single person, its theoretical application with multiple individuals will be discussed in Section 4. When operating without additional inputs, the method defaults to specific settings for segmentation, colors and other attributes. However, it is anticipated that the user might prefer specifying custom configurations, necessitating supplementary inputs such as segmentation prompts, color palettes, etc., as detailed in corresponding subsections below. The output of the method is a set of PNG images showing the resulting animated character and no background (where the alpha channel is set to zero).

The method consists of three independent processing steps. The main one is the segmentation and tracking of the different visual traits (hair, skin, clothes, etc.) of the person featured in the video frames. The second step focuses on rectifying segmentation errors through mask purging and simplifying their polygonal contours. The third and final step involves applying customizable finishing details, including colors, faux rim light, facial features and pixelation. Figure 2 shows the overall workflow of the method. Each component is elaborated upon in the subsequent subsections.

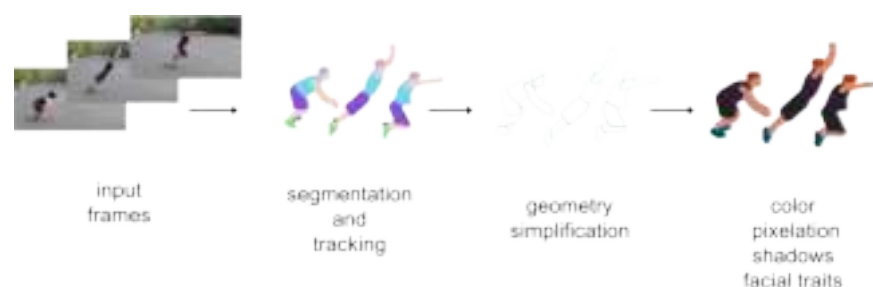


Figure 2. Overview of the system's workflow (inputs on the left, output on the right).

3.2. Segmentation and Tracking

In order to segment the different visual traits of the character the Segment Anything Model (SAM) [1] is used. SAM is a promptable segmentation model which is becoming a foundation model to solve a variety of tasks. Given an input image with three channels (height $\times$ width $\times$ 3) and a prompt, that can include spatial or text information, it generates a mask (height $\times$ width $\times$ 3), with each pixel assigned an RGB color. Each color represents an object label from the object label space $L = l_1, l_2, \dots$, inferred from the prompt, corresponding to the predicted object type at the corresponding pixel location in the input image. The prompt can be any information indicating what to segment in an image (e.g., a set of foreground/background points, a rough box or mask or free-form text). SAM consists of an image encoder (based on MAE [24] a pre-trained Vision Transformer [25]) that computes an image embedding, a prompt encoder (being the free-form text based on CLIP [26]) that embeds prompts, and then the two information sources are combined in a mask decoder (based on Transformer segmentation models [27]) that predicts segmentation masks.

Applying SAM at the frame level would result in different labels for the same object in different frames. SAM does not start from a closed list of object labels but rather creates and assigns labels during execution. Therefore, there is no guarantee that the same labels are assigned to the same object in each frame. To do it consistently, we use SAM-Track [2]. SAM results for the first frame are propagated to the second frame, with the same labels, with DeAOT [28], a method of hierarchical propagation for semi-supervised video object segmentation. The hierarchical propagation can gradually propagate information from past frames to the current frame and transfer the current frame feature from object-agnostic to object-specific. SAM is applied to subsequent frames to detect new objects, but the segmentation of already seen objects is carried out by DeAOT.

We have chosen keyword-based prompts to specify the desired segmentation. We provide some default prompts for common scenarios (e.g., 'hair', 'skin', 'shoes', 't-shirt', 'trousers') but it is anticipated that in many scenarios, the user will need to specify an alternative. These prompts are not processed with the free-text capabilities of SAM because of their limitations, instead they are processed with Grounding-DINO [29], an open-set object detector, which translate them into object boxes that are sent to SAM as box prompts. One problem we have identified is that certain object combinations lead to incorrect masks when they are specified in the same prompt. To solve this, we allow specifying multiple prompts that are processed independently. Figure 3 shows the workflow of the segmentation and tracking stage.

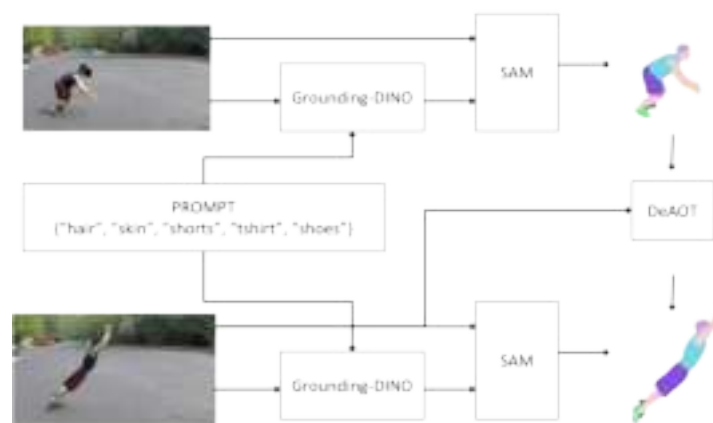


Figure 3. Workflow of the segmentation and tracking stage for the first two frames (inputs on the left, output on the right).

3.3. Contours Simplification

One of the desired choices for the target visual style is the possibility to display very simple geometries, i.e., shape contours with short polygonal chains, in order to mimic the visual aspect of, e.g., "Another World". To achieve it, a series of steps are carried out.

For each frame, the result of the segmentation and tracking step is a mask with each pixel assigned an RGB color representing an object label. The mask is first divided into submasks, one for each color/label. Each submask is divided, in turn, into contours with the algorithm from [30]. Each individual contour is represented with a vector of the vertex coordinates of the boundary line segments. The number of line segments of each contour is reduced with the Douglas–Peucker algorithm [3]. The algorithm starts by joining the endpoints of the contour with a straight line. Vertices closer to the line than a given tolerance value t are removed. Next, the line is split at the point farthest from the reference line, creating two new lines. The algorithm works recursively until all vertices within the tolerance are removed. Tolerance t is one of the optional parameters of the proposed method and let the user control the level of simplification. As each contour is simplified independently, undesirable spaces can appear between the geometries of bordering contours that previously were perfectly aligned. To avoid this, and before the simplification step, all contours are dilated convolving the submask with a 4×4 kernel. It is worth mentioning that the simplification algorithm may fail for certain contour geometries and values of t . In that situations (usually very small contours) the original contour is kept. See Algorithm 1 and Figure 4.

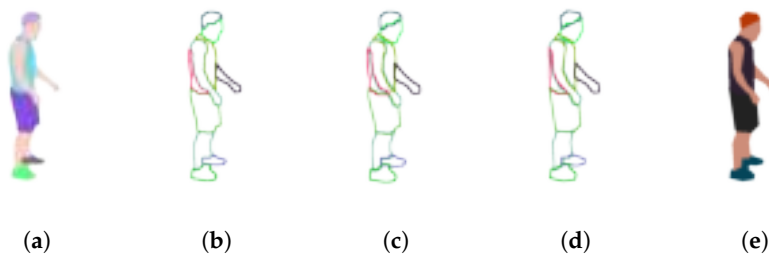


Figure 4. Graphical overview of the contour simplification stage of one frame: (a) result of the segmentation and tracking step; (b) contours detection; (c) contours dilation; (d) contours simplification; (e) resulting colored frame.

Algorithm 1 Contour simplification algorithm

Require: M a list containing submasks, α a minimum contour area, t a tolerance value for simplification

Ensure: R a list of contours where $R_{i,j}$ is the simplified contour j of submask i

```

1: function SIMPLIFY( $M, \alpha, t$ )
2:   for  $i \leftarrow 0$  to  $\text{length}(M) - 1$  do
3:      $C_i \leftarrow \text{getContours}(M_i)$ 
4:     for  $j \leftarrow 0$  to  $\text{length}(C_i) - 1$  do
5:       if  $\text{areaOf}(C_{i,j}) > \alpha$  then
6:          $D_{i,j} \leftarrow \text{dilateContour}(C_{i,j})$ 
7:         if  $\text{contourCanBeSimplified}(D_{i,j}, t)$  then
8:            $R_{i,j} \leftarrow \text{simplifyContour}(D_{i,j}, t)$ 
9:         else
10:           $R_{i,j} \leftarrow D_{i,j}$ 
11:        end if
12:      end if
13:    end for
14:  end for
15:  return  $R$ 
16: end function
17:
18:  $\text{contourCanBeSimplified}$  returns false if the Douglas–Peucker algorithm fails, as may
    happen for certain contour geometries and values of  $t$ .

```

3.4. Finishing Details

The third and final step of the method entails applying customizable finishing details, granting users flexibility in shaping the visual style of the outcome. The most important aspect is the color. The method generates a visual guide with unique identifiers for each color segment, as shown in Figure 5. One of the inputs to the method is a color palette, which indicates the RGB values that will be applied to each segment identifier.

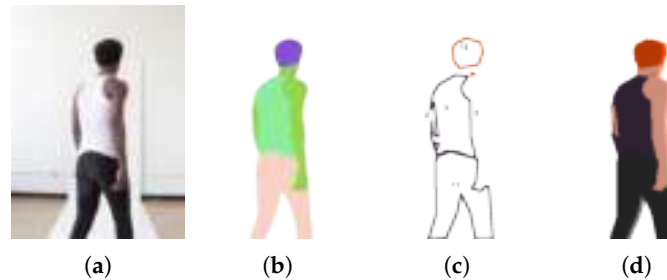


Figure 5. Coloring process: (a) input frame; (b) segmentation; (c) visual guide with unique identifiers for each color segment; (d) final frame.

Another finishing detail, which is optional, is the possibility of adding some schematic-looking facial features. The features are generated from the 68 facial landmarks that can be obtained using the frontal face detector described in [31], which employs a Histogram of Oriented Gradients (HOG) [32] combined with a linear classifier, an image pyramid and a sliding window detection scheme. Another option is the possibility of adding a very simple rim light effect by superimposing the same image with different brightness and displaced horizontally. Finally, a pixelation effect is achieved by resizing the image down using bilinear interpolation and scaling it back up with nearest neighbor interpolation. Figure 6 shows how rim light, facial features and pixelation are progressively applied to an example frame.



Figure 6. Optional finishing details: (a) Flat method result (b) Rim light (c) Facial features (d) Pixelation.

4. Experiments and Results

In order to evaluate the performance of the method, we collected a set of 25 short videos ranging from 5 to 20 s in duration. The videos showcase a lone human subject displaying various clothing styles and engaging in diverse actions. The videos were obtained from the UCF101 Human Actions dataset [33], the Fashion Video Dataset [34], the Kinetics dataset [35] and YouTube. The assembled dataset includes both low- and high-resolution videos and both in-the-wild and controlled-setting videos. The dataset and the results of the experiments can be found in [36]. Figures 7 and 8 show some of example results.

To assess the perceptual quality of the generated animations, a user study was conducted. A survey was designed comprising questions about three visual aspects: shape quality, color consistency and temporal consistency, plus and overall score. Scoring options were based on the Category Rating (ACR) described in ITU-T P.910 recommendation [37] scale from 1 = Bad to 5 = Excellent). The subjective quality assessment of the results is summarized in Table 1. We report the MOS (mean opinion score described in ITU-T P.800.1 [38]). The videos are grouped into three categories, in-the-wild low resolution

(ITW-LR), in-the-wild high resolution (ITW-HR) and controlled setting high resolution (C-HR).

Table 1. Subjective quality assessment MOS values of the ACR (1 = Bad to 5 = Excellent) of shape quality, color consistency, temporal consistency and overall quality. ^a ITW-LR → in-the-wild low resolution; ^b ITW-HR → in-the-wild high resolution; ^c C-HR → controlled setting high resolution.

Subset	#	Shape	Color	Temporal	Overall
ITW-LR ^a	5	3.20	3.24	3.84	3.36
ITW-HR ^b	14	3.81	4.03	4.33	3.99
C-HR ^c	6	4.70	4.60	4.67	4.67
all		3.90	4.01	4.31	4.02

Strengths: The results show that the method is able to synthesize quality animations from diverse input videos (an average MOS 4.02 over 5). The performance significantly improves (4.67) on high-resolution videos taken in controlled conditions (videos 2, 11, 12, 13, 14 and 15). The method exhibits a very good temporal consistency and can correctly process videos with different poses and appearances, dynamic shots (e.g., video 5), videos with partial shots (e.g., the medium close-up shot in video 7) and diverse backgrounds.

It is worth noting that, although it is not the primary focus of this work, the method can process videos containing multiple people. Objects associated with different individuals are tracked independently, each with a unique identifier, even if the objects are of the same type. This enables mapping these objects to different colors, allowing the entire pipeline to be applied to multi-person shots. Figure 9 shows an example result from a multi-person frame.

Limitations: The segmentation and tracking performance decreases in videos with low resolution (all videos from the UCF101 Human Actions dataset), complex backgrounds (videos 17 and 20), low-contrast colors (videos 8, 10, 22 and 23), inappropriate lighting (videos 14 and 25) and occlusions (video 23). Figure 10 shows four example frames with errors from four different videos (videos 22, 23, 25 and 8 from the test dataset).

Computational cost: The method has been implemented utilizing the Python programming language. The experiments have been run on a high-end server with a quadcore Intel i7-3820 at 3.6 GHz with 64 GB of DDR3 RAM, and 4 NVIDIA Tesla K40 GPU cards with 12 GB of GDDR5 each. The peak RAM usage of the method was recorded at 3.1 GB during the execution of the segmentation task, which also accounts for a peak GPU RAM usage of 2.1 GB. For a single frame, step 1 (segmentation and tracking) takes 0.56 s, step 2 (contours simplification) takes 0.43 s and step 3 (finishing details, mainly facial features) takes 1.32 s. Overall, the method operates at 0.43 FPS (1 FPS without facial features).

Comparison with image-to-image generative models: Although we are not aware of other methods that have attempted to solve the problem for this specific visual style, an alternative could involve using conditional generative models. One potential pipeline, described in [13], adapts Stable Diffusion [7] for image-to-image translation using ControlNet [10], and employs tiled inputs along with video-guided style interpolation (EbSynth [14]) to maintain visual consistency across frames. We tested this processing pipeline with some videos from the test dataset to evaluate the feasibility of this approach. To achieve the desired style, we fine-tuned Stable Diffusion with Dreambooth [39]. Figure 11 shows four example frames obtained with this approach, but to assess the temporal consistency the results can be found in our results repository in [36]. The results reveal several drawbacks compared to the proposed method. Firstly, they exhibit poor temporal consistency across frames. Secondly, the method requires specifying the level of fidelity to the desired style, where higher fidelity compromises the original frame structure and details, leading to an undesirable trade-off. Thirdly, the performance varies significantly depending on various aspects of the input video, such as scene and resolution. Lastly, the method operates significantly slower than the proposed method, with a processing speed of only 0.3 FPS.

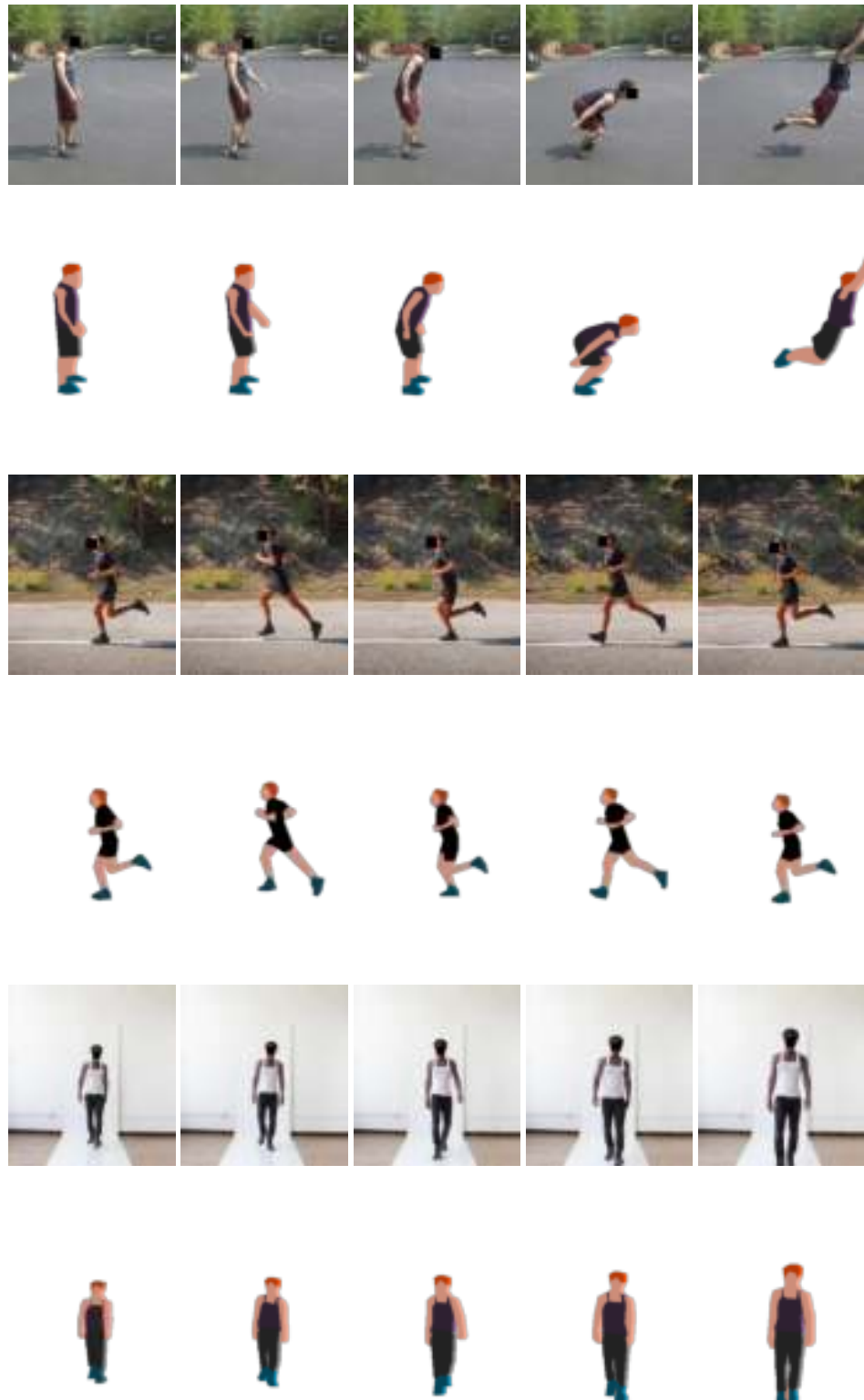


Figure 7. Example results. Odd rows: input videos (faces have been anonymized). Even rows: method results.

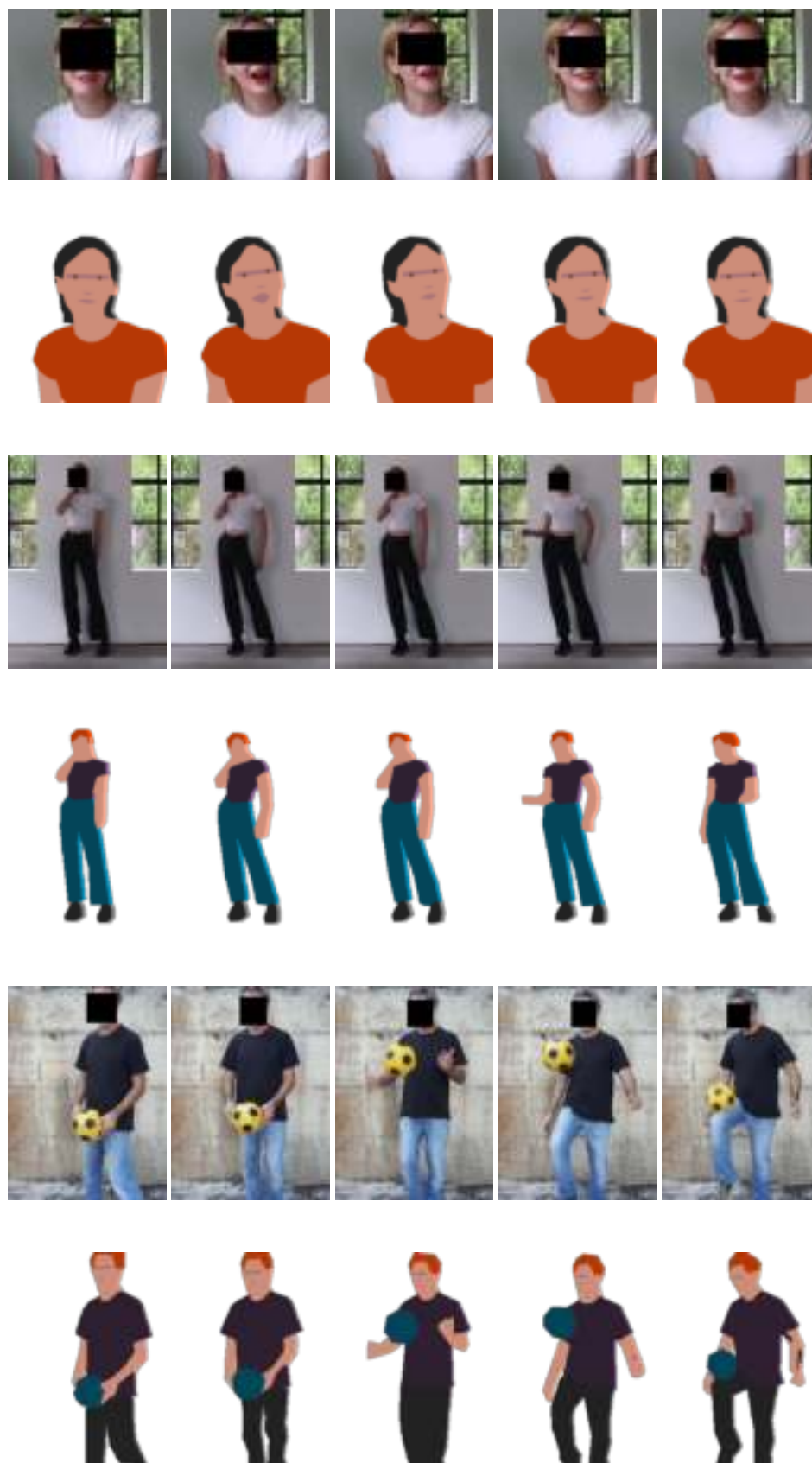


Figure 8. Example results. Odd rows: input videos (faces have been anonymized). Even rows: method results.

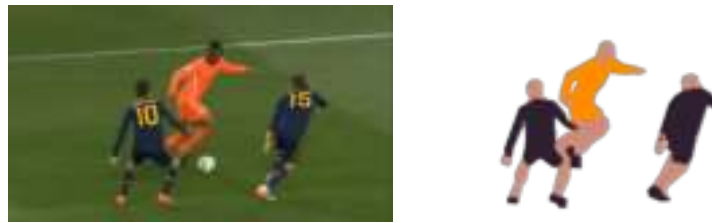


Figure 9. Example synthesized animation frame (right) and its related original frame from an example input multi-person video.



Figure 10. Example failure cases. Top row: input videos (videos 22, 23, 25 and 8 from the test dataset, faces have been anonymized); bottom row: results.



Figure 11. Comparison with a generative pipeline: (a) input frame; (b) generative pipeline described in [13]; (c) Lester (proposed method).

Discussion: The results obtained are very satisfactory and confirm that the latest advances in segmentation and tracking make new applications possible, also in the field of image synthesis. The proposed method has proven to be an effective method for generating rotoscopic animations from videos. The limitations detected do not represent a handicap since it is expected that the method will be used in videos selected or created expressly. Nor does its computational cost since the method is expected to be used in editing tasks that do not require real-time execution. The method is primarily designed for videos featuring a single person, but could be adapted to videos with multiple people by using multiple parameterizations for segmentation prompts and color palettes.

5. Conclusions

In this work we present Lester, a method for automatically synthesizing 2D retro-style animated characters from videos. Rather than addressing the challenge with an image-to-image generative model, or with human pose estimation tools, it is addressed mainly as an object segmentation and tracking problem. Subjective quality assessment shows that the approach is feasible and that the method can generate quality animations from videos with different poses and appearances, dynamic shots, partial shots and diverse backgrounds. Although it is not the objective, it is worth mentioning that the segmentation and tracking performance of the method is less consistent in the wild, especially against low-resolution videos. But, the method is expected to be used on selected or custom-made videos in semi-controlled environments.

Due to its capabilities, the method constitutes an effective tool that significantly reduces the cost of generating rotoscopic animations for video games or animated films. Due to its characteristics, it is simpler and less deterministic than generative models-based pipelines and more practical and flexible than techniques based on human pose estimation.

Funding: This research is partially funded by the Spanish Ministry of Science and Innovation under contract PID2019-107255GB, and by the SGR programme 2021-SGR-00478 of the Catalan Government.

Data Availability Statement: The dataset and the results of the experiments are publicly available at <https://github.com/rtous/lester> (accessed on 20 July 2024). The code is available at <https://github.com/rtous/lester-code> (accessed on 20 July 2024).

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ACR	Absolute Category Rating
GAN	Generative Adversarial Networks
HOG	Histogram of Oriented Gradients
MOS	Mean Opinion Score
SAM	Segment Anything Model

References

1. Kirillov, A.; Mintun, E.; Ravi, N.; Mao, H.; Rolland, C.; Gustafson, L.; Xiao, T.; Whitehead, S.; Berg, A.C.; Lo, W.Y.; et al. Segment anything. *arXiv* **2023**, arXiv:2304.02643.
2. Cheng, Y.; Li, L.; Xu, Y.; Li, X.; Yang, Z.; Wang, W.; Yang, Y. Segment and Track Anything. *arXiv* **2023**, arXiv:2305.06558.
3. Douglas, D.H.; Peucker, T.K. Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or its Caricature. In *Classics in Cartography: Reflections on Influential Articles from Cartographica*; John Wiley & Sons: Hoboken, NJ, USA, 2011; pp. 15–28. [\[CrossRef\]](#)
4. Chen, Y.; Lai, Y.K.; Liu, Y.J. CartoonGAN: Generative Adversarial Networks for Photo Cartoonization. In Proceedings of the CVPR. IEEE Computer Society, Salt Lake City, UT, USA, 18–23 June 2018; pp. 9465–9474.
5. Chen, J.; Liu, G.; Chen, X. AnimeGAN: A Novel Lightweight GAN for Photo Animation. In *Artificial Intelligence Algorithms and Applications, Proceedings of the 11th International Symposium, ISICA 2019, Guangzhou, China, 16–17 November 2019*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 242–256. [\[CrossRef\]](#)

6. Liu, Z.; Li, L.; Jiang, H.; Jin, X.; Tu, D.; Wang, S.; Zha, Z. Unsupervised Coherent Video Cartoonization with Perceptual Motion Consistency. In Proceedings of the The Thirty-Sixth AAAI Conference on Artificial Intelligence (AAAI-22), Virtual, 22 February–1 March 2022. [\[CrossRef\]](#)
7. Rombach, R.; Blattmann, A.; Lorenz, D.; Esser, P.; Ommer, B. High-Resolution Image Synthesis with Latent Diffusion Models. *arXiv* **2021**, arXiv:2112.10752.
8. Esser, P.; Kulal, S.; Blattmann, A.; Entezari, R.; Muller, J.; Saini, H.; Levi, Y.; Lorenz, D.; Sauer, A.; Boesel, F.; et al. Scaling Rectified Flow Transformers for High-Resolution Image Synthesis. *arXiv* **2024**, arXiv:2403.03206.
9. Meng, C.; He, Y.; Song, Y.; Song, J.; Wu, J.; Zhu, J.Y.; Ermon, S. SDEdit: Guided Image Synthesis and Editing with Stochastic Differential Equations. In Proceedings of the International Conference on Learning Representations, Virtual, 25–29 April 2022.
10. Zhang, L.; Rao, A.; Agrawala, M. Adding Conditional Control to Text-to-Image Diffusion Models. In Proceedings of the 2023 IEEE/CVF International Conference on Computer Vision (ICCV), Paris, France, 2–6 October 2023.
11. Saharia, C.; Ho, J.; Chan, W.; Salimans, T.; Fleet, D.J.; Norouzi, M. Image super-resolution via iterative refinement. *arXiv* **2021**, arXiv:2104.07636.
12. Saharia, C.; Chan, W.; Chang, H.; Lee, C.; Ho, J.; Salimans, T.; Fleet, D.; Norouzi, M. Palette: Image-to-Image Diffusion Models. In Proceedings of the ACM SIGGRAPH 2022 Conference Proceedings, New York, NY, USA, 7–11 August 2022; SIGGRAPH '22. [\[CrossRef\]](#)
13. Yang, S.; Zhou, Y.; Liu, Z.; Loy, C.C. Rerender A Video: Zero-Shot Text-Guided Video-to-Video Translation. In Proceedings of the ACM SIGGRAPH Asia Conference Proceedings, Sydney, NSW, Australia, 12–15 December 2023.
14. Jamriska, O. ebsynth: Fast Example-Based Image Synthesis and Style Transfer. 2018. Available online: <https://github.com/jamriska/ebsynth> (accessed on 20 July 2024).
15. Brooks, T.; Peebles, B.; Holmes, C.; DePue, W.; Guo, Y.; Jing, L.; Schnurr, D.; Taylor, J.; Luhman, T.; Luhman, E.; et al. Video Generation Models as World Simulators. 2024. Available online: <https://openai.com/index/video-generation-models-as-world-simulators/> (accessed on 20 July 2024).
16. Chen, Z.; Li, S.; Haque, M. An Overview of OpenAI's Sora and Its Potential for Physics Engine Free Games and Virtual Reality. *EAI Endorsed Trans. Robot.* **2024**, *3*. [\[CrossRef\]](#)
17. Liang, D.; Liu, Y.; Huang, Q.; Zhu, G.; Jiang, S.; Zhang, Z.; Gao, W. Video2Cartoon: Generating 3D cartoon from broadcast soccer video. In Proceedings of the MULTIMEDIA '05, Singapore, 11 November 2005.
18. Ngo, V.; Cai, J. Converting 2D soccer video to 3D cartoon. In Proceedings of the 2008 10th International Conference on Control, Automation, Robotics and Vision, Hanoi, Vietnam, 17–20 December 2008; pp. 103–107. [\[CrossRef\]](#)
19. Weng, C.Y.; Curless, B.; Kemelmacher-Shlizerman, I. Photo Wake-Up: 3D Character Animation From a Single Photo. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019.
20. Tous, R. Pictonaut: Movie cartoonization using 3D human pose estimation and GANs. *Multimed. Tools Appl.* **2023**, *82*, 1–15. [\[CrossRef\]](#)
21. Ruben Tous, J.N.; Igual, L. Human pose completion in partial body camera shots. *J. Exp. Theor. Artif. Intell.* **2023**, 1–11. [\[CrossRef\]](#)
22. Fišer, J.; Jamriska, O.; Simons, D.; Shechtman, E.; Lu, J.; Asente, P.; Lukáč, M.; Šýkora, D. Example-Based Synthesis of Stylized Facial Animations. *ACM Trans. Graph.* **2017**, *36*, 1–11. [\[CrossRef\]](#)
23. Gatys, L.A.; Ecker, A.S.; Bethge, M. A Neural Algorithm of Artistic Style. *arXiv* **2015**, arXiv:1508.06576.
24. He, K.; Chen, X.; Xie, S.; Li, Y.; Dollar, P.; Girshick, R. Masked Autoencoders Are Scalable Vision Learners. In Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 18–24 June 2022; pp. 15979–15988. [\[CrossRef\]](#)
25. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In Proceedings of the 9th International Conference on Learning Representations, ICLR 2021, Virtual, 3–7 May 2021. Available online: <https://openreview.net/> (accessed on 20 July 2024).
26. Radford, A.; Kim, J.W.; Hallacy, C.; Ramesh, A.; Goh, G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J.; et al. Learning Transferable Visual Models From Natural Language Supervision. In Proceedings of the 38th International Conference on Machine Learning, Virtual, 18–24 July 2021; pp. 8748–8763.
27. Carion, N.; Massa, F.; Synnaeve, G.; Usunier, N.; Kirillov, A.; Zagoruyko, S. End-to-End Object Detection with Transformers. In Proceedings of the Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, 23–28 August 2020; pp. 213–229. [\[CrossRef\]](#)
28. Yang, Z.; Yang, Y. Decoupling Features in Hierarchical Propagation for Video Object Segmentation. In Proceedings of the Advances in Neural Information Processing Systems (NeurIPS), New Orleans, LA, USA, 28 November–9 December 2022.
29. Liu, S.; Zeng, Z.; Ren, T.; Li, F.; Zhang, H.; Yang, J.; Li, C.; Yang, J.; Su, H.; Zhu, J.; et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv* **2023**, arXiv:2303.05499.
30. Suzuki, S.; be, K. Topological structural analysis of digitized binary images by border following. *Comput. Vision Graph. Image Process.* **1985**, *30*, 32–46. [\[CrossRef\]](#)
31. Kazemi, V.; Sullivan, J. One millisecond face alignment with an ensemble of regression trees. In Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 1867–1874. [\[CrossRef\]](#)

32. Dalal, N.; Triggs, B. Histograms of oriented gradients for human detection. In Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA, 20–26 June 2005; Volume 1, pp. 886–893. [[CrossRef](#)]
33. Soomro, K.; Zamir, A.R.; Shah, M. UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild. *arXiv* **2012**, arXiv:1212.0402.
34. Zablotskaia, P.; Siarohin, A.; Zhao, B.; Sigal, L. DwNet: Dense warp-based network for pose-guided human video generation. In Proceedings of the 30th British Machine Vision Conference 2019, BMVC 2019, Cardiff, UK, 9–12 September 2019; p. 51.
35. Kay, W.; Carreira, J.; Simonyan, K.; Zhang, B.; Hillier, C.; Vijayanarasimhan, S.; Viola, F.; Green, T.; Back, T.; Natsev, P.; et al. The Kinetics Human Action Video Dataset. *arXiv* **2017**, arXiv:1705.06950.
36. Tous, R. Lester Project Home Page. Available online: <https://github.com/rtous/lester> (accessed on 20 July 2024).
37. ITU. Subjective Video Quality Assessment Methods for Multimedia Applications. ITU-T Recommendation P.910. 2008. Available online: <https://www.itu.int/rec/T-REC-P.910> (accessed on 20 July 2024).
38. ITU. Mean Opinion Score (MOS) Terminology. ITU-T Recommendation P.800.1 Methods for Objective and Subjective Assessment of Speech and Video Quality. 2016. Available online: <https://www.itu.int/rec/T-REC-P.800.1> (accessed on 20 July 2024).
39. Ruiz, N.; Li, Y.; Jampani, V.; Pritch, Y.; Rubinstein, M.; Aberman, K. DreamBooth: Fine Tuning Text-to-Image Diffusion Models for Subject-Driven Generation. *arXiv* **2023**, arXiv:2208.12242.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.

Article

A Swarm Intelligence Solution for the Multi-Vehicle Profitable Pickup and Delivery Problem

Abeer I. Alhujaylan ^{1,*} and Manar I. Hosny ² 

¹ Department of Information Technology, College of Computer, Qassim University, Buraydah 51452, Saudi Arabia

² Computer Science Department, College of Computer and Information Sciences, King Saud University, Riyadh 11362, Saudi Arabia; mifawzi@ksu.edu.sa

* Correspondence: a.alhujaylan@qu.edu.sa

Abstract: Delivery apps are experiencing significant growth, requiring efficient algorithms to coordinate transportation and generate profits. One problem that considers the goals of delivery apps is the multi-vehicle profitable pickup and delivery problem (MVPPDP). In this paper, we propose eight new metaheuristics to improve the initial solutions for the MVPPDP based on the well-known swarm intelligence algorithm, Artificial Bee Colony (ABC): K-means-GRASP-ABC(C)S1, K-means-GRASP-ABC(C)S2, Modified K-means-GRASP-ABC(C)S1, Modified K-means-GRASP-ABC(C)S2, ACO-GRASP-ABC(C)S1, ACO-GRASP-ABC(C)S2, ABC(S1), and ABC(S2). All methods achieved superior performance in most instances in terms of processing time. For example, for 250 customers, the average times of the algorithms was 75.9, 72.86, 79.17, 73.85, 76.60, 66.29, 177.07, and 196.09, which were faster than those of the state-of-the-art methods that took 300 s. Moreover, all proposed algorithms performed well on small-size instances in terms of profit by achieving thirteen new best solutions and five equal solutions to the best-known solutions. However, the algorithms slightly lag behind in medium- and large-sized instances due to the greedy randomised strategy and GRASP that have been used in the scout bee phase. Moreover, our algorithms prioritise minimal solutions and iterations for rapid processing time in daily m-commerce apps, while reducing iteration counts and population sizes reduces the likelihood of obtaining good solution quality.



Citation: Alhujaylan, A.I.; Hosny, M.I. A Swarm Intelligence Solution for the Multi-Vehicle Profitable Pickup and Delivery Problem. *Algorithms* **2024**, *17*, 331. <https://doi.org/10.3390/a17080331>

Academic Editor: Vittorio Maniezzo

Received: 2 July 2024

Revised: 17 July 2024

Accepted: 25 July 2024

Published: 31 July 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

Keywords: multi-vehicle profitable pickup and delivery problem; Artificial Bee Colony; swarm intelligence; metaheuristic algorithms

1. Introduction

Over the past five years, delivery apps have experienced substantial growth, particularly food delivery apps, where COVID-19 propelled the industry forward a few years as millions of individuals in lockdown ordered food and other essential services online. The grocery delivery platform called Instacart reported that its objectives for 2022 were met during the third week of lockdown. In 2022, China emerged as the largest food delivery market, with a staggering market value of USD 42.5 billion. By 2029, the market size of the food delivery app industry as a whole is anticipated to reach USD 165 billion [1]. Moreover, 85% of Americans place online food orders via food ordering applications. Moreover, 80% of restaurants in the United States provide customers with the convenience of mobile ordering [2]. Similarly, food delivery applications in Saudi Arabia are providing instantaneous services and a streamlined user experience for individuals to enjoy food. As a result, new applications that vie to provide services, convenience, and comfort to over 35 million individuals emerge periodically. Examples of these applications that have become very popular in Saudi Arabia are Mrsool, Uber Eats, Talabat, Hunger station, Ninja, Jahez, To You, The Chifz, Shgardi, and NANA [3]. The escalating demand for food delivery applications has concurrently heightened the strain on delivery enterprises, which require

expeditious algorithms to coordinate the transportation of goods in order to generate profits and guarantee client contentment. One of the problems that was presented in the literature that mimics the goal of food delivery apps, increasing profits, achieving user satisfaction, and reducing costs is the multi-vehicle profitable pickup and delivery problem (MVPPDP).

Gansterer et al. [4] introduced the MVPPDP, which is a subtype of the selective pickup and delivery problem (SPDP) and a novel variation of the well-known pickup and delivery problem (PDP). Within the constraints of a one-day planning horizon, the MVPPDP seeks to identify the most profitable routes by minimising travel expenses and maximising revenue collection. The MVPPDP is a static problem characterised by a standardised fleet of vehicles, a central depot, a collection of customers, and a predetermined quantity of requests and products. A customer pair, consisting of the pickup customer and the delivery customer, defines each request. By doing so, the products are conveyed from a pickup customer to a delivery customer in order to generate a revenue stream for the operation. MVPPDP real-world applications include mobile applications for food delivery services and delivery companies.

The MVPPDP is one of the combinatorial optimisation (CO) problem applications that receives particular attention from researchers in computer science and operations research. Solving a CO problem mainly requires searching to find the optimal or best possible solution from a limited set of feasible candidate solutions. Numerous algorithms can be used to solve CO problems, and these algorithms can be classified into two categories: exact algorithms, which are guaranteed to obtain optimal solutions in an adequate amount of time, and approximate algorithms, usually called heuristics or metaheuristics, which are not guaranteed to deliver an optimal solution but provide a good solution in a reasonable amount of time. The words heuristic and meta belong to the Greek language, where heuristic is from *heuriskein*, meaning *‘to find’* or *‘to discover’* new methods and strategies to solve problems, while the prefix meta means *‘beyond’*, in an upper level. Metaheuristic methods can be classified based on the number of solutions used at the same time into two categories: single solution-based metaheuristics (S-metaheuristics) and population-based metaheuristics (P-metaheuristics). In S-metaheuristics, the focus is on modifying and improving a single candidate solution. On the other hand, P-metaheuristics preserve and improve several candidate solutions instead of just a single solution. They are based on iteratively improving solutions in a population and evolving the whole population until a set of good solutions is reached. Exploration-oriented features of P-metaheuristics have made them better than S-metaheuristics at diversifying the search space. The most famous examples of P-metaheuristics are algorithms that are inspired by natural behaviour such as genetic algorithms and swarm intelligence (SI) [5].

Generally, there are two phases that should be applied to obtain a good problem solution: solution construction and solution improvement. Solution construction involves creating one or more initial possible solutions that can be considered as a starting point for exploring the search space. Solution improvement is based on modifying the starting solutions to improve them, according to predefined strategies, until a high-quality solution is reached or stopping criteria have been met (e.g., a maximum number of iterations, or a specific amount of time) [5].

Due to the fact that the MVPPDP is an NP-hard problem (as it is a subtype of the SPDP, which is a special case of PDP) [4], metaheuristic algorithms can be employed to efficiently locate optimal solutions in a reasonable amount of time, particularly for problems of moderate to large sizes. In this paper, we introduce novel algorithms for tackling this problem, with the objective of enhancing the initial solution proposed in [6,7]. Our approach is based on the utilisation of the Artificial Bee Colony (ABC) algorithm, which is one of the widely recognised artificial intelligence algorithms known as SI. Diverse strategies were implemented with ABC, each demonstrating efficacy in accelerating the search procedure and reducing processing time.

The primary goals of this paper are (1) assisting food delivery applications and delivery companies in optimising their services through accelerated customer selection and

route planning in order to maximise profits as well as customer satisfaction; (2) efficient transport planning that can help conserve energy resources by reducing the environmentally damaging effects of gas emissions; and that (3) for some tested instances, the proposed approaches achieve results that surpass the state-of-the-art methods documented in the literature for the MVPPDP.

The rest of this paper is organised as follows: a review of some related work is presented in Section 2. Section 3 introduces the formal problem definition. The proposed method is described in detail in Section 4. Section 5 illustrates the experimental results. Finally, conclusions and future work are presented in Section 6.

2. Related Studies

The MVPPDP is classified as a type of SPDP and its objective is to maximise profits while minimising travel expenses by visiting only profitable customer pairs. Achieving a favourable route that maximises the difference between the total revenue earned and the total travelling cost is the objective of both the MVPPDP and the profitable tour problem (PTP). The primary distinction is that in the PTP, the vehicle route is not subject to any constraints, including but not limited to the maximum trip time, vehicle capacity limits, or precedence constraints [8]. Research focusing on the PTP is in short supply in the scientific literature [9]. Hence, we commence by presenting research pertaining to the MVPPDP.

Gansterer et al. [4] introduced the initial four algorithms. The variable neighbourhood search (VNS) metaheuristic was implemented in [10]. The greedy construction heuristic (C1) and the two-stage cheapest insertion heuristic (C2) were employed during the construction phase of the VNS. During the improvement phase, the subsequent neighbourhood operators were implemented: the pairwise backward exchange, the pairwise forward exchange, forward insertion, relocate pairs, backward insertion, inter-tour exchange, inter-tour insert, inter-dummy insert, inter-dummy exchange and the gravity centre exchange. The authors implemented two approaches for applying the neighbourhood operators: a sequential (Seq) method, which employs a predetermined sequence of neighbourhoods, and a self-adaptive (Sea) method, which adjusts itself based on the search status to determine the sequence of neighbourhoods. In order to compare the proposed method with an alternative algorithm, a guided local search (GLS) metaheuristic-based algorithm was ultimately implemented. An evaluation of the proposed algorithm was conducted on 36 randomised instances of varying sizes. In terms of solution quality, the experimental results demonstrated that both variants of the VNS outperformed the GLS.

Haddad [11] introduced a new algorithm to solve the MVPPDP. Pending a solution to the profitable pickup and delivery problem, they proposed an algorithm known as IPPD, which was derived from a combination of an iterated local search and random variable neighbourhood descent. During the search, the IPPD algorithm was not restricted to accepting only viable solutions. For the construction of the initial solution, the C1 heuristic utilised by Gansterer et al. [4] was implemented. To enhance the solution, several neighbourhood moves were implemented: pair swap or pair shift, block swap or block shift, pickup or delivery shift, inter-block swap or inter-block shift, gravity centre exchange, insert operator and remove operator, and inter-pair swap or inter-pair shift. On the benchmark instances suggested by Gansterer et al. [4], the proposed algorithm was evaluated. It demonstrated its effectiveness in handling small- and medium-sized cases by identifying new optimal solutions for six cases. However, for large instances, the performance of the proposed algorithm was inadequate.

In Alhujaylan and Hosny [6], the construction phase of the widely recognised greedy randomised adaptive search procedure (GRASP) was employed to develop preliminary solutions for the MVPPDP. They used the same dataset that was created by [4]. They conducted a performance comparison between the method and two construction heuristics that had been previously documented in the literature for constructing initial solutions to the MVPPDP. The experimental results demonstrated that the proposed method outperformed

alternative construction heuristics, particularly in instances of a small size, where eight new best initial solutions to the problem were obtained.

In [7], Alhujaylan and Hosny presented new heuristics to construct initial solutions for the MVPPDP. The heuristics are based on first clustering the search space of the MVPPDP using three clustering methods: K-means, adaptive K-means, and ant colony optimisation (ACO). Then, a modified version of the GRASP has been used to construct the initial MVPPDP solution based on the results of each clustering algorithm. The proposed solution was applied in the dataset that was presented by the researchers of [4]. They compared their results with those from the other construction heuristics that have been previously used for the MVPPDP. The experimental results proved the effectiveness of the proposed algorithms in terms of both solution quality and processing time.

In [12], the authors incorporated time windows with uncertain travel times into the MVPPDP. The researchers utilised a subset of the database created by [4], which comprised twenty and fifty requests, corresponding to two and three vehicles, respectively. The objective of the client is to identify a solution that optimises the net profit, which is calculated as the difference between the revenue collected, the cost of the route, and the expense incurred due to time window violations. They took wide time windows and standard time windows into consideration. Under the assumption that travel times are independent Gamma-distributed random variables and with a risk-neutral perspective, they demonstrated that the anticipated costs of punctuality and early arrival can be calculated in a closed form. The authors suggested employing a graph search approach as the solution method. Experiments have demonstrated that precise solutions can be obtained in a reasonable amount of time.

3. Problem Definition

The MVPPDP is a predefined static problem having several constituents. It is postulated that a centralised depot is responsible for receiving customer requests. These requests comprise pickup and delivery customers in pairs. To fulfil these demands, we operate a fleet of identical vehicles that convey a variety of identical products from a subset of pickup customers to their respective delivery customers, thereby generating a specific profit. The following limitations of the MVPPDP must be considered:

- The pairing constraint stipulates that a predefined customer pair is assigned to each request (pickup and delivery).
- Precedence constraint: Priority must be given to visiting the pickup customer over the delivery customer.
- The time limit for journeys: Every vehicle is subject to a daily travel time restriction that must not be surpassed during the service of customers.

One limitation is the capacity of each vehicle, which must not be surpassed during product collection from pickup customers.

- Every vehicle should depart from and arrive at the depot with an empty load.
- A single visit is permitted per customer.

The MVPPDP can be formally defined as follows [10]: Let $G = (V, A)$ be a graph in which $V = \{0, \dots, 2n + 1\}$ defines the vertex set, where the vertex $(0, 2n + 1)$ corresponds with a central depot, while the remaining vertices represent the pickup customers $P = \{1, \dots, n\}$, and delivery customers $D = \{n + 1, \dots, 2n\}$. $A = \{(i, j) : i, j \in V, i \neq j\}$ defines the arc set, where each arc is associated with a non-negative routing cost c_{ij} . Each delivery vertex i has a revenue r_i to be gained when visiting it. Also, each vertex i has a supply q_i (pickup, $q_i > 0$) or demand (delivery, $q_{n+i} = -q_i$). At the start, there is no supply or demand in the depot ($q_0 = 0$). In addition, each vertex i has service duration d_i . There is a set of vehicles $K = \{1, \dots, m\}$, and each vehicle has a load capacity C and maximum tour time T .

The objective function (OF) of the MVPPDP aims to maximise the total profit by subtracting the total travel costs from the total revenues. It can be formulated as follows:

$$\text{Maximize} \sum_{i \in V} \sum_{j \in V} \sum_{k \in K} (r_i - c_{ij}) x_{ijk}, \quad (1)$$

The mathematical model of the MVPPDP can be formulated as in [4,6] as follows:

$$\sum_{i \in V} \sum_{k \in K} x_{ijk} \leq 1 \quad j \in V \quad (2)$$

$$\sum_{j \in V} \sum_{k \in K} x_{ijk} \leq 1 \quad i \in V \quad (3)$$

$$x_{i0k} = 0 \quad i \in V, k \in K \quad (4)$$

$$x_{2n+1,jk} = 0 \quad j \in V, k \in K \quad (5)$$

$$\sum_{i \in V} (x_{ijk} - x_{jik}) = 0 \quad j \in V \setminus \{0, 2n+1\}, k \in K \quad (6)$$

$$\sum_{j \in V} (x_{ijk} - x_{n+i,jk}) = 0 \quad i \in P, k \in K \quad (7)$$

$$\sum_{j \in V} x_{0jk} = \sum_{i \in V} x_{i,2n+1,k} = 1 \quad k \in K \quad (8)$$

$$(x_{ijk} = 1) \Rightarrow Q_{jk} = Q_{ik} + q_j \quad i \in V, j \in V \setminus \{0\}, k \in K \quad (9)$$

$$Q_{ik} \leq C \quad i \in V, k \in K \quad (10)$$

$$Q_{0k} = 0 \quad k \in K \quad (11)$$

$$B_{ik} \leq B_{n+i,k} \quad i \in P \quad (12)$$

$$B_{jk} \geq X_{ijk} (B_{ik} + d_i + t_{ij}) \quad i \in V, j \in V, k \in K \quad (13)$$

$$B_{2n+1,k} \leq T \quad k \in K \quad (14)$$

$$B_{0k} = 0 \quad k \in K \quad (15)$$

$$x_{ijk} \in 0, 1 \quad i \in V, j \in V, k \in K \quad (16)$$

$$Q_{ik}, B_{ik} \geq 0 \quad i \in V, k \in K \quad (17)$$

Constraints (2) and (3) require that each vertex be visited at most once. Constraints (4) and (5) prevent any vehicle from entering the origin depot (beginning point) or leaving the destination depot (end point). Constraint (6) indicates flow conservation. Constraint (7) states that each client pair (pickup and delivery) must be served by the same vehicle. Constraint (8) requires that all vehicles start and return to the depot. Constraint (9) states that exceeding the vehicle capacity is not permitted. Constraint (10) specifies that the load of a vehicle is limited by C . Constraint (11) specifies that all vehicles start with an empty cargo. Constraint (12) indicates that the delivery customer cannot be visited before the pickup customer. Constraint (13) states that the earliest time to begin service at a vertex j is determined by adding the beginning of the service time at the vertex i to the service time at and the travel time between i and j . Constraint (14) indicates that the maximum time for each vehicle is limited by T . Constraint (15) specifies that each vehicle begins at the depot at time zero. Finally, decision variables are defined by (16) and (17).

4. Proposed Method

In the current study, we employed the Artificial Bee Colony (ABC) algorithm that was first proposed by Karaboga [13]. It mimics the collaborative and collective foraging behaviour of a bee colony. In the ABC algorithm, there are three different types of bees:

scout bees, employed bees, and onlooker bees. Each type has a different role in the exploration and exploitation of food sources. A food source represents the solution to a problem, and better solutions belong to food sources with more nectar. The ABC algorithm works as follows: Initially, the scout bees are assigned randomly to the search space to find initial food sources. Then, the employed bees are sent out to exploit the places discovered by scout bees, with each employed bee assigned to one food source. After that, the employed bees start a neighbourhood search to find a better food source nearby. After finding a better food source, the employed bees leave all previous food sources and exploit only the better ones. These employed bees then return to the hive and share their information about discovered food sources with onlooker bees through what is called the waggle dance. The decision of whether the onlooker bees will follow the employed bees depends on the richness of the food sources. If the onlooker bees do decide to follow, they then become employed bees and repeat the procedure of the employed bees. After several iterations, the food source might be exhausted; at that point, the employed bees become scout bees and start searching randomly to find a new food source to replace the previous one. The flowchart of the ABC algorithm is illustrated in Figure 1 [14].

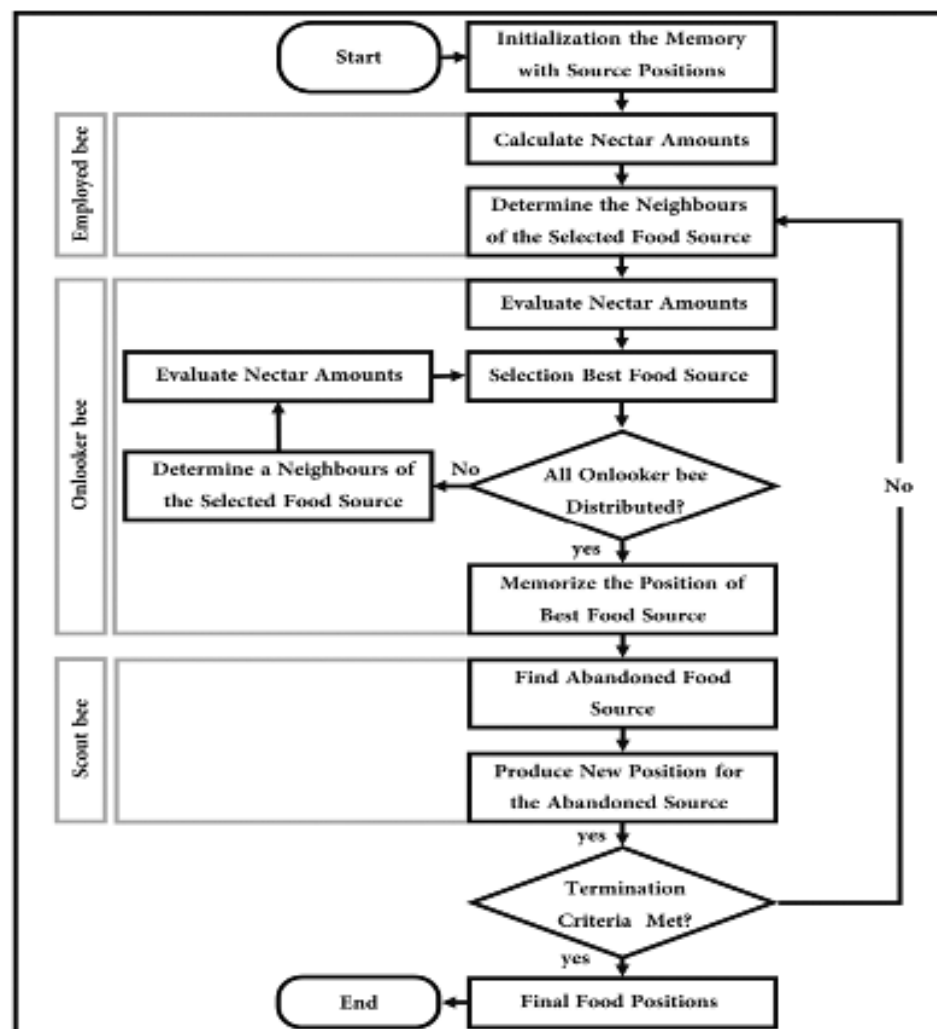


Figure 1. Flowchart of the Artificial Bee Colony algorithm [14].

The ABC heuristic has been used for solving several VRP variants, such as being used by Szeto et al. [15] to solve the capacitated vehicle routing problem, by Wu et al. [16] to solve the VRP with time windows, and by Yao et al. [17] to solve the periodic VRP.

In this paper, we aim to improve the initial solutions proposed in [7]. There are many metaheuristics that have been used to enhance the quality of solutions, such as local search (LS), simulated annealing (SA), and VNS. In our study, we also use a metaheuristic that combines greedy and randomised searches and is based on honeybee behaviour, namely the ABC. The main reason for selecting the ABC rather than other honeybee algorithms is its popularity, with the number of studies that have used the ABC compared to other variants reaching 54%, as reported in Karaboga's survey [18]. For tackling the MVPPDP, two versions of the ABC are proposed: ABC without clustering (ABC) and ABC with clustering (ABC(C)), as will be explained next.

4.1. ABC without Clustering (ABC)

In this part of the research, we aimed to improve the initial solutions that have been achieved using the improved GRASP in [7]. The ABC metaheuristic consists of three phases: the employed bee phase, the onlooker bee phase, and the scout bee phase. The main goal of the first two phases is to increase the intensification of the search space to obtain good solutions, while the last phase aims to increase the diversification and prevent getting stuck in local optima. The details of our proposed algorithm are illustrated below.

1. Employed bee phase:

After assigning the initial solutions to the employed bees, they start a neighbourhood search to find better solutions nearby by randomly selecting one of a set of neighbourhood operators. The neighbourhood operators aim to make a small change to the solution to improve it. Thus, for each solution, one route is selected randomly to apply one operator that is also selected randomly. The set of neighbourhood operators is described below:

- Relocate a pickup customer only: Choose one pickup customer randomly and relocate its position, such that the new pickup position must not come after its delivery customer.
- Relocate a delivery customer only: Choose one delivery customer randomly and relocate its position, such that the new delivery position must not come before its pickup customer.
- Relocate a customer pair: Choose a customer pair randomly and relocate their positions, preserving the precedence constraint (i.e., pickup before delivery).
- Swap two customer pairs: Select two customer pairs randomly and swap their positions.
- Two-Opt operator: Select two consecutive pairs of customers (a consecutive pair is one in which a pickup customer is followed directly by its delivery customer) and reorder the edges between pairs while considering the precedence constraint.
- Remove one customer pair randomly and insert an unvisited one: Remove a customer pair randomly, select an unvisited customer pair that has a high insertion ratio (IR) using Equation (18), and insert it into the best position in the route.

$$IR_{customerpair} = \frac{r_{customerpair}}{c_{(depot,PC)}}, \quad (18)$$

where r is the revenue of the customer pair and $c_{(depot,PC)}$ is the distance between the depot and the pickup customer of that pair.

- Remove a customer pair with a low IR and insert an unvisited pair with a high IR: Compute the IR for all visited pairs using Equation (18), and remove a customer pair with the lowest IR value. Then, recompute the IR for all unvisited customer pairs, select one with the highest IR value, and insert it into the best position in the route.
- Insert a customer pair with a high IR: Compute the IR for all unvisited customer pairs, select one with the highest IR value, and insert it into the best position in the route.

After applying the selected neighbourhood operator, if the new solution S' is feasible, the fitness of S' is computed using Equation (1) and compared with the fitness of the old solution S . To escape from local optima, we decided to use the demon algorithm (DA) rather

than a simple local search algorithm, since the DA has the ability to accept a feasible non-improved new solution. The DA is a variant of the SA that was presented by Creutz [19]. The acceptance function of the DA is based on the value of the demon (credit) that is initialised at the beginning with a given value D . In the case where the new solution S' is not an improvement, the difference (ΔE) between the fitness of S and S' is compared with the demon value, and S' is accepted if this difference is less than or equal to D . Then, if S' is accepted, it is memorised, and the value of D is decreased; otherwise, the old solution S is memorised, the D value is increased, and a non-improvement counter for this solution is increased by one. The DA is simpler and faster than the SA because the latter requires more computational time for generating a random number and an exponential function. Also, it contains only one parameter that needs to be tuned. The DA proved to be an effective algorithm when it was used to solve several real-life problems, achieving good results in terms of the quality of solution and time, when compared with the SA [19].

2. Onlooker bee phase:

Only the solutions that have good OF values have a high probability of being assigned to onlooker agents. Several strategies have been proposed for selecting solutions, such as roulette wheel selection, tournament selection, stochastic universal sampling, and rank-based selection [20]. We adopted the most popular one, which is the roulette wheel selection strategy, in which the probability of selection P_i is computed as follows:

$$P_i = \frac{OF_i}{\sum_{j=1}^n OF_j}, \quad (19)$$

where OF_i is the objective function for a solution i , and $\sum_{j=1}^n OF_j$ is the sum of objective function values for all solutions in a population. After a solution is selected, one of the solution routes is selected randomly and all the neighbourhood operators used in the employed bee phase are applied sequentially to it, and then the fitness of the new solution is measured using Equation (1). To increase the intensification process and focus only on promising areas of the search space, the acceptance criterion is based on a straightforward LS (local search) algorithm. That is, if S' is better than S , memorise S' , and there is no need to complete the remaining operators; otherwise, try the next operator. If the old solution is not improved by any operator, the old solution is memorised, and a non-improvement counter for this solution is increased by one.

3. Scout bee phase:

The scout bee phase aims to increase the diversification and expand the solution space, thus preventing getting stuck in local optima. The scout bee phase starts when a non-improvement counter for a solution S_i reaches a predefined number of trials, called a limit. The limit refers to the number of attempts that were made to improve the solution S_i , in both employed and onlooker bee phases, respectively. Three important actions should be performed when a non-improvement counter for solution S reaches the limit: (1) clear the non-improvement counter, (2) clear the solution S_i and (3) reconstruct the solution S_i . A suitable reconstruction strategy has an important role in the scout bee phase, where it either contributes to enhancing the quality of the solution or leaves the solution unchanged. Thus, we proposed two strategies to reconstruct the solution S_i . The details of these reconstruction heuristics are presented as follows:

- First strategy (ABC(S1)): One of the following three heuristics is selected randomly:

1. Random insertion heuristic:

While the time constraint is not violated, the customer pairs are selected randomly and inserted directly into the route. Each pickup customer is followed by its delivery customer; therefore, there is no need to check the vehicle's capacity and precedence constraints.

2. Greedy and randomised heuristics:

While the time constraint is not violated, the customer pairs are selected randomly and inserted into the best positions in the route. In addition to the time constraint, the vehicle capacity and precedence constraints must not be violated.

3. Greedy insertion heuristic:

The insertion ratio is computed for all customer pairs using Equation (18). The customer pair that has the highest IR value is then inserted into the route in the best position. Time, vehicle capacity, and precedence constraints must not be violated.

- Second strategy (ABC(S2)): This strategy works exactly like the GRASP in [7].

After the solution S_i is reconstructed, the OF is computed using Equation (1), and S_i is memorised as a new solution and used to start again with the improvement process using the employed and onlooker bee phases.

The outline of the routing phase of our ABC is presented in Algorithm 1, where each notation is defined as follows: Pop Size represents the population size; Max Iter: the maximum number of iterations; Num Vehicles: the quantity of vehicles; US: pairs of customers who have not been served; ΔE : the difference between OF values for an old and new solution; D : demon value; and *Non-improvement*: a counter to count the number of unsuccessful attempts to improve the solution.

Algorithm 1: The pseudocode of the proposed ABC.

```

Pop = { $S_1, S_2, S_3, \dots, S_n$ }; /* Initial solutions that have been generated using GRASP(V2)* /
Limit = initial limit value,  $D$  = initial demon value, Non-improvement = 0, Global_Best = Best
solution in Pop;
For  $M = 1$  to  $Max\_Iter$ 
  /** Employed bee phase **/
  For  $i = 1$  to Pop_Size
    Select randomly one route of solution  $S_i$ ;
    Apply one randomly selected neighbourhood operator to the current route of solution
     $S_i$  to obtain a new solution  $S'_i$ ;
    If  $S'_i$  is feasible Then Compute the OF of  $S'_i$ ;
    End If
    If ( $OF(S'_i) > OF(S_i)$ )
      Memorise the new solution  $S'_i$ ;
    Else if ( $\Delta E \leq D_i$ )
      Memorise the new solution  $S'_i$ ;
      update the demon value:  $D_i = D_i - \Delta E$ ;
    Else
      Memorise the old solution  $S_i$ ;
      update the demon value:  $D_i = D_i + \Delta E$ ;
      Non-improvement( $i$ ) = Non-improvement( $i$ ) + 1;
    End If
  End For
  /** Onlooker bee phase **/
  Select  $N$  solutions from the employed bees using roulette wheel selection strategy;
  For  $i = 1$  to  $N$ 
    Select randomly one route of solution  $S_i$ ;
    For  $j = 1$  to number of neighbourhood operators
      Apply a selected neighbourhood operator ( $j$ ) to the current route of solution
       $S_i$  to obtain a new solution  $S'_i$ ;
      If  $S'_i$  is feasible Then Compute the OF of  $S'_i$ ;
      End If
      If ( $OF(S'_i) > OF(S_i)$ )
        Memorise the new solution  $S'_i$ ;
        Break;
      End if
    End For
  End For

```

```

If the solution  $S_i$  has not improved after applying the neighbourhood operators
Non-improvement( $i$ ) = Non-improvement( $i$ ) + 1;
End If
End For
/*** Scout bee phase ***/
For  $i = 1$  to  $N$ 
If Non-improvement( $i$ ) > Limit
Non-improvement( $i$ ) = 0;
 $S_i = 0$ ;
Reconstruct  $S_i$  by using either the S1 or S2 strategy of scout bee phase; Compute the OF for  $S_i$ ;
End if
End For
Assign the solution that has maximum OF to the Best_Solution;
If (Global_Best < Best_Solution)
Global_Best = Best_Solution;
End If
End For
Output: the Global_Best solution

```

4.2. ABC with Clustering (ABC(C))

In this part of the study, the ABC method described in Section 4.1 was used to improve the initial solutions that were constructed using the various GRASP(C) methods reported in [7], namely K-means-GRASP(C), modified-K-means GRASP(C), and ACO-GRASP(C). Again, there are some modifications that must be considered when using the ABC with those algorithms. When using the following operators, the insertion process should be restricted to the customer pairs inside the same cluster only: remove one customer pair randomly and insert an unvisited one, remove a customer pair with a low IR and insert an unvisited pair with a high IR, and insert a customer pair with a high IR. Also, the insertion restriction should be applied in the scout bee phase in both strategies (S1 and S2). We used the same parameter values that have been used in the ABC (without clustering) to improve the initial solutions in this part of the study. Moreover, the same strategies from the scout bee phase (S1 and S2) have been used in the ABC(C) algorithms. Therefore, overall, we have six algorithms that use the ABC(C): (1) K-means-GRASP(C)-ABC(C)S1, (2) K-means-GRASP(C)-ABC(C)S2, (3) modified K-means-GRASP(C)-ABC(C)S1, (4) modified K-means-GRASP(C)-ABC(C)S2, (5) ACO-GRASP(C)-ABC(C)S1, and 6) ACO-GRASP(C)-ABC(C)S2. For example, the first algorithm refers to generating the initial solution using the GRASP after clustering the search space by using the K-means algorithm and then improving it using the ABC with reconstruction strategy S1.

5. Computational Experiments

All algorithms have been implemented in MATLAB (R2017b) on a laptop powered by an Intel Core i7-4510U processor operating at 2.00 GHz (2601 MHz, two cores, four logical processors). The following subsections provide descriptions of the datasets utilised and the specifics of parameter tuning prior to presenting the experimental results.

5.1. Test Instances

To test the performance of our proposed methods, we used the same instances used by [4]. The dataset comprises 36 instances, which have been categorised into three distinct groups: small size (consisting of 20 and 50 customers, respectively, served by two and three vehicles), medium size (comprising 100 and 250 customers, served by 4 and 5 vehicles, respectively), and large size (500 and 1000 customers served by 6 and 8 vehicles, respectively). Moreover, twelve instances comprise each group. These occurrences exhibit variability in terms of both time and revenue. To generate short and long routes, the total time limit is either small or large, with a range of 2500 to 15,000 s.

Furthermore, the revenue amounts are established in one of three ways: at random, proportional to demand, or equal for all customers. Revenue is acquired from the delivery customer subsequent to the shipment of merchandise originating from the pickup customer. The quantity of goods is specified as a positive integer between 1 and 50.

5.2. Parameter Tuning

5.2.1. Parameter Tuning of ABC without Clustering

The ABC algorithm has three parameters: the number of iterations (Max_Iter), the size of the population (Pop_Size), and the initial value of the limit. Additionally, since we used the DA in the employed bee phase, the initial value of the demon (D) needs to be tuned. The details of testing each parameter are presented below.

Number of Iterations

Each data instance of 100 customers was tested with different numbers of iterations: 10, 50, 100, 500, and 750. As seen in Table 1, the results of testing showed that there was no enhancement in the objective function values after 500 iterations. Thus, the maximum number of iterations was taken to be 500.

Table 1. Results of tuning the number of iterations of the ABC.

Dataset Instances	Number of Iterations				
	10	50	100	500	750
13-100-F-S	32,274.04	35,432.98	39,437.98	39,438.58	39,438.58
14-100-F-L	63,311.86	63,311.86	63,311.86	63,455.13	63,455.13
15-100-P-S	68,625.48	68,625.48	71,185.28	72,032.22	72,032.22
16-100-P-L	115,792.8	115,792.8	124,608.2	131,404.9	131,404.9
17-100-R-S	67,974.56	74,246.27	74,246.27	74,330.95	74,330.95
18-100-R-L	112,986.9	112,986.9	116,775.2	119,991.9	119,991.9

Size of Population

The same dataset instances were tested again to select the appropriate number of solutions in the population. At the beginning, the population size was set to 30, because it provided the best results given in [7]. Then, we increased the population to different sizes: 40, 50, and 60 solutions. Table 2 presents the values of the OF after setting the number of iterations to 500. The results showed that increasing the population size to more than 50 did not lead to an improvement in the objective function value. Thus, the population size was taken to be 50.

Table 2. Results of tuning the population size of the ABC.

Dataset Instances	Population Size			
	30	40	50	60
13-100-F-S	39,438.58	39,438.58	41,472.86	41,472.86
14-100-F-L	63,455.13	66,453.66	66,453.66	66,453.66
15-100-P-S	72,032.22	72,032.22	72,170.54	72,170.54
16-100-P-L	131,404.9	131,404.9	131,404.9	131,404.9
17-100-R-S	74,330.95	77,484.06	77,484.06	77,484.06
18-100-R-L	119,991.9	120,249.7	126,504.3	126,504.3

Limit Value

The limit value of the non-improvement counter is another important parameter of the ABC algorithm because it affects the conversion of the search process. Thus, the value selected for the limit should be suitable for the number of iterations. Therefore, we tested the limit using two values: 100 and 200. The results in Table 3 show that increasing the

limit value to more than 100 did not contribute to an improvement in the objective function value. Thus, the limit value was taken to be 100.

Table 3. Results of tuning the limit value of the ABC.

Dataset Instances	Limit	
	100	200
13-100-F-S	41,472.86	41,472.86
14-100-F-L	66,453.66	66,453.66
15-100-P-S	72,170.54	72,170.54
16-100-P-L	131,404.9	131,404.9
17-100-R-S	77,484.06	77,484.06
18-100-R-L	126,504.3	126,504.3

Demon Value

Several values have been tested to select the best initial value of the demon (D): 1000, 2500, 5000, and 7500. The results in Table 4 indicated no improvement in objective function if the demon value was greater than 5000. Thus, the demon value D was taken to be 5000.

Table 4. Results of tuning the demon value of the ABC.

Dataset Instances	Demon Value			
	1000	2500	5000	7500
13-100-F-S	41,472.86	41,596.33	41,954.86	41,954.86
14-100-F-L	66,453.66	66,683.84	66,683.84	66,683.84
15-100-P-S	72,170.54	72,449.56	72,449.56	72,449.56
16-100-P-L	131,404.9	131,404.9	131,404.9	131,404.9
17-100-R-S	77,484.06	77,484.06	77,484.06	77,484.06
18-100-R-L	126,504.3	126,504.3	126,504.3	126,504.3

5.3. Parameter Tuning of ABC(C)

Since we used the same ABC algorithm, with little modification, in the ABC with the clustering approach, we adopted the same parameter values that have been used in the ABC for our ABC(C).

5.4. Experimental Results

As previously mentioned, we proposed two versions of the ABC to improve the initial solutions: the ABC used with GRASP, and the ABC(C) used with the clustering methods, i.e., K-means-GRASP(C), modified K-means-GRASP(C), and ACO-GRASP(C). We also have two separate strategies for the scout bee phase. S1 constructs the solution by randomly selecting one of three strategies: random insertion, greedy, and randomised insertion or greedy insertion. After testing this strategy, we found the random insertion heuristic did not improve the results compared to others, so we eliminated this choice and kept the other two. The second strategy, S2, works like the GRASP in [7]. Therefore, we have eight overall proposed methods named as follows: (1) ABC(S1), (2) ABC(S2), (3) K-means-GRASP(C)-ABC(C)S1, (4) K-means-GRASP(C)-ABC(C)S2, (5) modified K-means-GRASP(C)-ABC(C)S1, (6) modified K-means-GRASP(C)-ABC(C)S2, (7) ACO-GRASP(C)-ABC(C)S1, and (8) ACO-GRASP(C)-ABC(C)S2. All proposed algorithms were run five times, as performed with the previous algorithms in [6] and [7]. We compared our proposed algorithms and the state-of-the-art algorithms presented in [10,11], which are C1-Seq VNS, C2-Seq VNS, C1-Self Adaptive VNS, C2-Self Adaptive VNS and IPPD, for solving the MVPPDP in terms of the processing time and profits gained.

First, with respect to the processing time, all previous algorithms have been run for 5 min for each instance. We note, though, that this time is not considered acceptable for the m-commerce apps (e.g., Hungerstation, MRSOOL, etc.) that are used daily and rely on the

speed of responding, so in our case, we used the stopping condition of 500 iterations to measure the processing time. Table 5 presents the comparison results between our proposed algorithms and state-of-the-art algorithms in terms of time (in seconds). Also, the average of each category (highlighted rows in Table 5) has been used to compare the performance of the algorithms as shown in Figure 2. From Table 5 and Figure 2, it is evident that our proposed clustering-based algorithms outperform all previous algorithms with respect to processing time in all instances, except for instances involving 1000 customers. Also, the other proposed algorithms that are not based on clustering (ABC(S1) and ABC(S2)) were faster than the previous algorithms for small- and medium-sized instances. The reason why our algorithms do not excel in large-sized instances is probably due to the scout bee strategies that have been used, namely the greedy method in the first strategy and the GRASP in the second. In the greedy method, the customer pairs are selected based on a specific measure and then inserted into the best position that leads to decreasing the travel cost. This approach gives good results but consumes time, especially in large-sized instances. In the second strategy, the unvisited customer pairs are selected based on computing the IR, which also leads to consuming time in large-sized instances. Overall, though, the clustering of the search space clearly helped to process large-sized instances and made the processing time faster compared to other algorithms that deal with all customers without clustering.

Table 5. Comparison between state-of-the-art algorithms and the proposed algorithms based on time (seconds).

Dataset	Previous Algo- rithms(5 min Runtime)	K-Means- GRASP- ABC(C)S1	K-Means- GRASP- ABC(C)S2	Modified K-Means- GRASP- ABC(C)S1	Modified K-Means- GRASP- ABC(C)S2	ACO- GRASP- ABC(C)S1	ACO- GRASP- ABC(C)S2	ABC(S1)	ABC(S2)
1-20-F-S	300	14.48	14.47	15.20	14.58	16.78	12.12	31.04	30.65
2-20-F-L	300	12.13	12.42	13.82	13.17	13.51	11.45	30.20	30.56
3-20-P-S	300	16.62	17.48	17.83	17.17	18.72	15.85	33.77	39.28
4-20-P-L	300	12	12.01	12.61	12.24	15.63	12.98	24.43	28.45
5-20-R-S	300	14.87	15.21	14.97	14.57	16.67	14.57	30.79	31.17
6-20-R-L	300	11.32	12.38	13.47	13.71	13.63	12.25	22.62	21.56
Average-20	300	13.57	14	14.65	14.24	15.82	13.20	28.81	30.28
7-50-F-S	300	23.24	24.29	25.67	25.49	25.45	22.03	50.10	54.80
8-50-F-L	300	19.79	21.24	21.53	21.45	22.18	20.65	43.81	46.63
9-50-P-S	300	25.67	24.64	26.58	25.15	25.87	20.49	50.55	54.72
10-50-P-L	300	21.63	22.64	24.41	22.96	23.34	20.90	44.77	48.77
11-50-R-S	300	26.31	26.58	26.86	26.05	27.49	20.23	48.90	51.94
12-50-R-L	300	22.73	20.93	23.37	21.87	22.77	20.91	45.94	48.34
Average-50	300	23.23	23.39	24.74	23.83	24.52	20.87	47.34	50.86
13-100-F-S	300	37.87	33.57	36.58	35.10	36.89	33.36	80.65	83.81
14-100-F-L	300	31.42	33.67	34.59	34.38	35.55	33.14	73.90	81.74
15-100-P-S	300	31.03	32.03	36.98	35.55	36.75	33.23	76.39	82.69
16-100-P-L	300	33.54	34.33	36.50	34.57	36.20	32.41	74.19	82.93
17-100-R-S	300	38	35.92	37	35.81	38.19	34.03	82.59	86.96
18-100-R-L	300	36.19	35.37	34.38	33.34	37.49	33.63	77.02	81.01
Average-100	300	34.68	34.15	36.01	34.79	36.85	33.30	77.46	83.19
19-250-F-S	300	73.50	72.85	78.10	73.30	74.19	67.79	176.81	204.37
20-250-F-L	300	79.65	73.28	80.50	76.28	77.66	65.10	181.10	195
21-250-P-S	300	74.20	72.09	75.23	72.88	74.05	63.51	169.26	195.17
22-250-P-L	300	73.82	72.718	79.53	74.40	77.86	65.44	179	193.60
23-250-R-S	300	74.67	72.99	77.63	72.38	76.21	66.44	177.25	193.93
24-250-R-L	300	79.55	73.21	84.04	73.84	79.62	69.46	178.99	194.47
Average-250	300	75.90	72.86	79.17	73.85	76.60	66.29	177.07	196.09
25-500-F-S	300	163.89	155.32	172.74	158.36	162.15	142.52	388.62	474.16
26-500-F-L	300	167.03	155.73	177.18	160.25	170.25	148.98	415.25	466.38
27-500-P-S	300	160.91	146.27	172.04	158.45	162.76	140.62	386.55	465.87
28-500-P-L	300	168.60	144.23	180.26	158.74	169.89	141.34	435.34	464.62

Table 5. Cont.

Dataset	Previous Algorithms(5 min Runtime)	K-Means-GRASP-ABC(C)S1	K-Means-GRASP-ABC(C)S2	Modified K-Means-GRASP-ABC(C)S1	Modified K-Means-GRASP-ABC(C)S2	ACO-GRASP-ABC(C)S1	ACO-GRASP-ABC(C)S2	ABC(S1)	ABC(S2)
29-500-R-S	300	170.15	140.16	170.50	158.58	162.42	139.80	383.89	468.15
30-500-R-L	300	169.78	141.24	174.22	161.07	169.30	143.36	428.65	464.89
Average-500	300	166.73	147.16	174.49	159.24	166.13	142.77	406.38	467.35
31-1000-F-S	300	432.51	354.73	448.29	408.67	426.32	397.22	1040.22	1406.22
32-1000-F-L	300	439.85	360.11	447.72	423.01	461.18	430.54	1197.38	1438.43
33-1000-P-S	300	434.34	364.52	432.93	418.51	391.01	416.76	1128.14	1398.49
34-1000-P-L	300	448.06	378.57	440.70	435.69	389.93	430.39	1204.97	1416.64
35-1000-R-S	300	451.79	404.20	425.83	421.47	381.24	411.31	1110.80	1296.40
36-1000-R-L	300	420.15	410.26	446.23	418.03	393.29	420.10	1222.33	1286.07
Average-1000	300	437.78	378.73	440.28	420.90	407.16	417.72	1150.64	1373.71

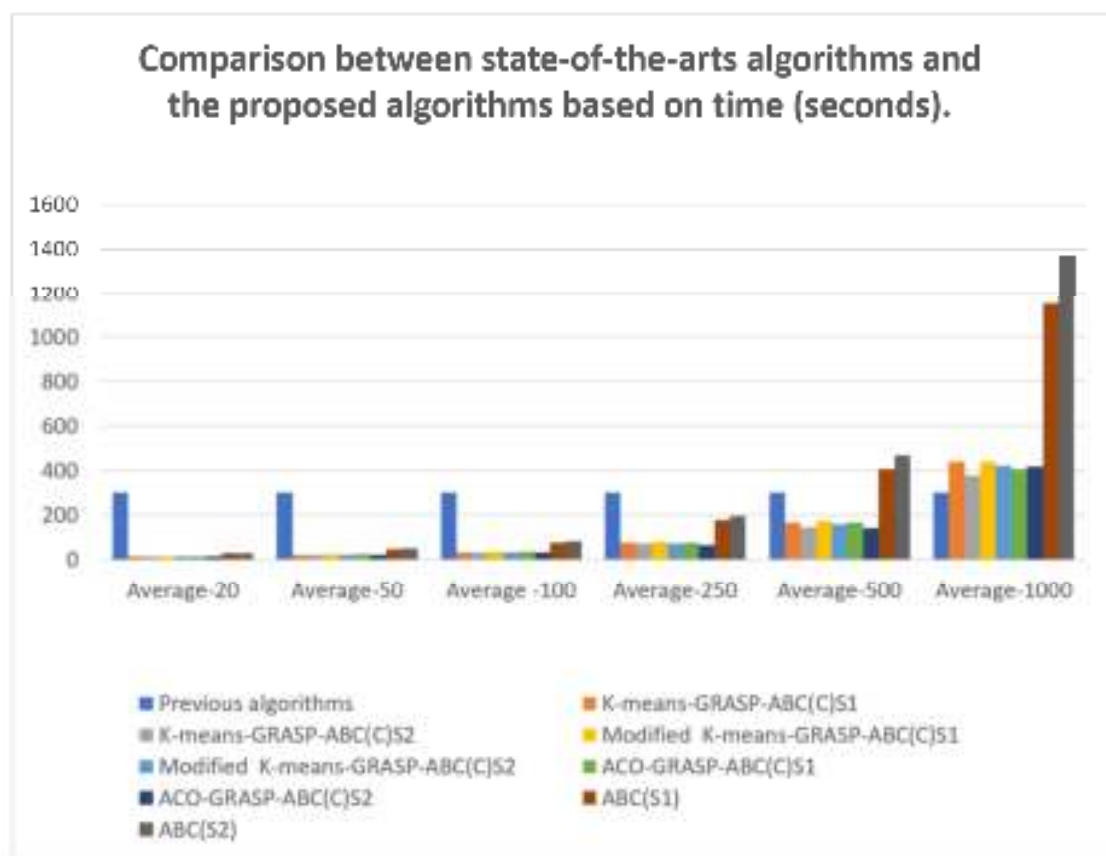


Figure 2. Comparison between state-of-the-arts algorithms and the proposed algorithms based on time (seconds).

On the other hand, we compared our proposed algorithms and the previous algorithms in terms of profit as computed in Equation (1). Table 6 presents the average profit for five runs, while Table 7 presents the best profit obtained over five runs and the gap between the best profit and the best known profit using the following equation:

$$Gap = \frac{Best\ profit\ over\ five\ runs - Best\ known\ profit}{Best\ known\ profit} * 100 \quad (20)$$

Table 6. Comparison of our proposed algorithms with previous methods based on average profit.

Dataset	<i>C1—Seq VNS [9]</i>	<i>C2—Seq VND [9]</i>	<i>C1—Self Adaptive VND [9]</i>	<i>C2—Self Adaptive VND [9]</i>	<i>IPPD [10]</i>	<i>K-Means- GRASP- ABC(C)S1</i>	<i>K-Means- GRASP- ABC(C)S2</i>	<i>Modified K-Means- GRASP- ABC(C)S1</i>	<i>Modified K-Means- GRASP- ABC(C)S2</i>	<i>ACO- GRASP- ABC(C)S1</i>	<i>ACO- GRASP- ABC(C)S2</i>	<i>ABC(S1)</i>	<i>ABC(S2)</i>
1-20-F-S	30,315.5	30,315.5	30,315.5	30,315.5	30,315.5	30,267.75	30,427.58	27,007.58	27,007.58	25,490.76	25,490.76	30,395.92	30,294.77
2-20-F-L	38,765.6	38,765.6	37,866.3	38,765.6	39,179.66	37,642.38	37,468.82	39,773.34	39,240.95	36,342.53	36,322.33	39,459.15	36,153.93
3-20-P-S	51,473.4	52,128.4	52,565.1	52,565.1	52,566	51,481.8	43,340.71	51,481.8	43,009.55	42,917.34	34,107.64	52,684.98	52,565.1
4-20-P-L	58,365.6	58,365.6	58,365.6	58,365.6	58,777	57,350.35	57,086.36	57,350.35	57,301.88	56,523.32	47,367.54	58,888.64	56,808.21
5-20-R-S	36,469.5	36,469.5	36,469.5	36,469.5	36,469.5	34,090.58	34,090.58	30,237.02	29,843.58	31,741.08	31,741.08	36,469.52	36,469.52
6-20-R-L	43,368.6	43,368.6	43,368.6	43,368.6	43,781.81	42,215.53	42,086.44	44,401.45	43,736.53	40,769.67	40,450.55	44,244.27	42,143.26
Average-20	43,126.37	43,235.53	43,158.43	43,308.32	43,514.91	42,174.73	40,750.08	41,708.59	40,023.35	38,964.12	35,913.32	43,690.41	42,405.8
Standard Deviation	10,289.94	10,399.10	10,559.14	10,475.04	10,560.13	10,436.42	9349.23	11,822.81	10,907.82	10,656.36	7500.01	10,573.40	10,312.25
7-50-F-S	22,306	22,306	22,306	22,306	22,307.15	18,662.83	21,603.83	18,662.83	21,603.83	21,281.88	25,123.15	24,975.25	24,998.45
8-50-F-L	58,050.8	58,050.8	57,303.4	57,996.7	58,050.8	50,772.64	50,873.68	50,881.55	50,773.54	46,699.31	47,534.79	53,555.25	53,268.82
9-50-P-S	57,651.7	57,651.7	57,651.7	57,651.7	57,651.7	58,996.17	53,258.83	58,996.17	53,064.14	55,643.25	55,643.25	57,651.71	57,651.71
10-50-P-L	132,648	132,648.0	128,353	128,334.8	129,640.3	109,726.8	117,770.8	116,372	116,149.7	97,740.02	106,491.9	124,819.3	128,477.1
11-50-R-S	33,340.4	33,340.4	33,340.4	33,340.4	33,340.4	28,865.48	28,907.33	28,865.48	28,907.33	31,251.34	29,969.16	33,054.57	33,098.58
12-50-R-L	79,350.3	79,350.3	77,655.5	79,350.3	77,832.56	76,422.62	76,503.28	76,309.46	76,408.91	64,359.55	64,519.79	76,582.15	75,732.86
Average-50	63,891.2	50,139.84	62,768.33	63,163.32	63,137.16	57,241.09	58,152.95	58,347.92	57,817.91	52,829.22	54,880.35	61,773.03	62,204.58
Standard Deviation	39,248.45	22,523.36	37,635.69	37,749.54	38,077.61	33,048.63	35,097.31	35,184.37	34,547.72	27,041.68	29,384.58	35,937.41	37,167.44
13-100-F-S	41,464.84	41,955.2	41,523.5	42,174	41,896.86	35,386.24	36,055.01	35,303.67	36,625.9	33,147.92	33,187.57	40,596.77	40,543.56
14-100-F-L	81,271.2	81,971	80,080.4	80,500.2	79,479.88	58,704.7	60,780.74	57,607.07	60,035.43	51,106.58	47,074.98	64,937.47	67,237.63
15-100-P-S	70,387.4	70,398.5	71,113.8	71,082.8	69,436.25	54,197	57,206.6	49,566.36	59,481.84	60,614.78	45,995.09	68,874.02	71,290.33
16-100-P-L	124,288	123,931	122,691.8	122,183.2	125,057	98,140.3	104,031.6	100,122.4	110,099	92,681.38	96,365.75	112,690.5	124,487.1
17-100-R-S	76,885.4	78,418.8	76,954.4	79,651.2	77,012.75	61,579.81	61,579.81	61,960.78	61,698.84	52,689.7	52,163.72	77,334.8	75,925.47
18-100-R-L	131,052.6	130,670.4	130,403.4	130,366.6	131,058.6	98,522.81	100,291.3	104,771.9	109,450.2	92,423.54	93,867.6	115,108.7	122,320.4
Average-100	87,558.24	87,890.82	87,127.88	87,659.67	87,323.56	67,755.14	69,990.84	68,222.03	72,898.54	63,777.32	61,442.45	79,923.7	83,634.09
Standard Deviation	34,099	33,674.36	33,546.8	33,090.45	34,316.01	25,383.15	26,636.59	28,060.86	30,021.67	24,032.04	26,834.7	29,022.89	33,175.38
19-250-F-S	73,259.94	72,445.3	72,703	72,287.9	70,799.19	36,045.94	32,341.35	43,729.29	38,692.81	34,038.31	33,942.57	48,988.62	46,216.19
20-250-F-L	132,996	132,438	132,394.4	133,404	129,485.3	54,144.64	63,245.12	70,325.74	81,370.79	51,239.48	57,273.9	71,397.91	81,288.1
21-250-P-S	111,334.4	111,817.8	109,449.4	109,355.8	110,440.3	62,245.05	71,066.44	74,931.5	83,992.55	72,198.2	69,410.97	86,624.32	93,568.56
22-250-P-L	197,899.8	197,214.2	194,131.2	194,549	194,338.6	112,454.5	131,917.5	133,567.4	155,750.3	95,906.32	127,789.5	146,104	160,301.3
23-250-R-S	162,410.8	162,448.2	158,164.4	159,129.6	158,414.6	99,900	90,042.33	109,717.8	93,812.47	102,500.1	99,734.68	125,201.2	120,776
24-250-R-L	269,072.8	268,713.8	266,559	267,291.4	266,204.2	148,156.7	176,579.9	180,807.7	177,336	146,270.4	164,012.2	204,037.4	203,859.8
Average-250	157,829	157,512.9	155,566.9	156,003	154,947	85,491.14	94,198.78	102,179.9	105,159.2	83,692.14	92,027.3	113,725.6	117,668.3
Standard Deviation	69,184.1	69,165.71	68,327.76	68,666.69	68,777.74	42,022.43	52,050.37	49,795.51	51,632.15	40,174	48,193.98	56,662.62	57,036.7

Table 6. *Cont.*

Dataset	C1—Seq VNS [9]	C2—Seq VND [9]	C1—Self Adaptive VND [9]	C2—Self Adaptive VND [9]	IPPD [10]	K-Means- GRASP- ABC(C)S1	K-Means- GRASP- ABC(C)S2	Modified K-Means- GRASP- ABC(C)S1	Modified K-Means- GRASP- ABC(C)S2	ACO- GRASP- ABC(C)S1	ACO- GRASP- ABC(C)S2	ABC(S1)	ABC(S2)
25-500-F-S	158,425.4	159,146.2	157,399.8	159,526.6	155,683.9	59,737.27	71,001.31	51,581.38	73,939.03	54,762.97	51,117.39	75,731.5	79,862.1
26-500-F-L	243,210	244,591.6	245,600.8	245,183.6	242,173.1	101,753.4	120,779.3	106,505.3	129,200.4	62,041.19	88,256.7	112,172	121,824.9
27-500-P-S	201,195.4	202,102.2	197,227.6	198,740	203,932.9	105,678.8	146,631.6	107,797.6	143,918.5	102,996.9	105,667.8	130,361.2	143,638.1
28-500-P-L	319,239.2	321,181.6	315,854.6	317,610.8	329,642.8	229,525.6	241,331.4	225,812.4	243,361.8	152,573.5	191,169.5	201,973.8	222,042.1
29-500-R-S	271,154.4	272,715.2	271,264.2	273,095	260,505.8	126,298.6	156,901.1	108,998.5	162,660.4	127,089.1	123,505.3	166,111.6	166,226.9
30-500-R-L	402,557.2	406,689.6	399,990.8	404,517	408,819.4	259,683.6	272,509	241,454	282,087	167,819.5	205,885.1	239,675.7	255,261.1
Average-500	265,963.6	267,737.7	264,556.3	266,445.5	266,793	147,112.9	168,192.3	140,358.2	172,527.9	111,213.8	127,600.3	154,337.6	164,809.2
Standard Deviation	69,184.1	69,165.71	68,327.76	68,666.69	68,777.74	42,022.43	52,050.37	49,795.51	51,632.15	40,174	48,193.98	56,662.62	57,036.7
31-1000-F-S	150,920.4	151,943.2	151,990	153,820.6	152,358.7	33,184.03	48,758.35	39,530.22	48,457.57	23,266.5	20,723.05	33,125.19	41,340.89
32-1000-F-L	218,211	218,213	223,999.8	222,072.8	222,089.1	92,970.35	93,386.23	92,142.15	95,500.88	24,110.85	44,359.35	57,137.24	68,705.69
33-1000-P-S	439,681.8	441,120.2	433,353.8	429,853	457,759.4	272,126.9	288,095.1	236,797.1	280,912.6	178,654.3	201,178.4	240,944.9	263,540.2
34-1000-P-L	678,396.2	651,754.6	677,511.6	669,212.8	681,370.5	498,835.3	519,115.5	492,753.1	507,105.4	346,244.9	366,004.4	397,930	426,372.5
35-1000-R-S	455,054.6	461,394	459,167.4	462,999.2	461,919.6	207,134	255,835.4	192,030.8	242,210.5	124,538.1	176,577.1	212,979.1	228,716
36-1000-R-L	668,655	662,605.4	668,615.4	667,184.8	676,159.7	464,549.3	450,834.3	407,703.7	448,210.5	297,212.8	322,568.7	364,643.1	372,491.4
Average-1000	435,153.2	431,171.7	435,773	434,190.5	441,942.8	261,466.7	276,004.1	243,492.9	270,399.6	165,671.2	188,568.5	217,793.2	233,527.8
Standard Deviation	219,981.4	212,861.8	218,443.1	216,320	221,371.6	190,345	187,196	176,803.2	183,520.7	135,712.2	140,506.7	151,318.2	155,901.2

Table 7. Comparison between the proposed algorithms and previous methods based on the best solution and gap values.

[illegible]

Table 7. Cont.

Dataset	Best Known Profit	C1—Seq VNS [4]		C2—Seq VND [4]		C1—Self Adaptive VND [4]		C2—Self Adaptive VND [4]		IPPD [10]		K-Means-GRASP-ABC(C)S1	
		Best Profit	Gap	Best Profit	Gap	Best Profit	Gap	Best Profit	Gap	Best Profit	Gap	Best Profit	Gap
9-50-P-S	57,651.7	57,651.7	0	57,651.7	0	57,651.7	0	57,651.7	0	57,651.7	0	58,996.17	2.3
10-50-P-L	132,648	132,648	0	132,648	0	132,648	0	132,648	0	132,648	0	114,962.9	−13.3
11-50-R-S	33,340.4	33,340.4	0	33,340.4	0	33,340.4	0	33,340.4	0	33,340.4	0	28,907.33	−13.2
12-50-R-L	79,350.3	79,350.3	0	79,350.3	0	79,350.3	0	79,350.3	0	77,832.56	−1.95	76,479.84	−3.6
Average-50	63,913.7	63,891.20	−0.1	63,891.20	−0.10	63,891.20	−0.10	63,891.20	−0.10	63,638.44	−0.43	58,174.87	−8.9
13-100-F-S	44,402.3	42,174.4	−5.3	42,174.4	−5.3	42,174.4	−5.3	42,215.5	−5.2	41,956.25	−5.83	36,322.94	−18.1
14-100-F-L	81,999.4	81,999.4	0	81,999.4	0	81,621	−0.5	80,848.1	−0.2	88,891.4	0	60,767.96	−25.8
15-100-P-S	71,512.4	70,836	−1	71,326.4	−0.3	71,399.2	−0.2	71,399.2	−0.2	71,234.58	−0.39	54,428.34	−23.8
16-100-P-L	125,032	124,770	−0.2	124,770	−0.2	123,766	−1	124,910	−0.1	125,786.7	−0.6	99,860.9	−20.1
17-100-R-S	80,878.8	80,878.8	0	78,818	−2.6	80,878.8	0	80,878.8	0	80,878.8	0	61,647.98	−23.7
18-100-R-L	131,845	131,473	−0.3	131,845	0	131,339	−0.4	131,339	−0.4	131,845	0	102,403.8	−22.3
Average-100	89,278.32	97,991.44	−1.13	88,488.87	−1.40	88,529.73	−1.23	88,598.43	−1.02	90,098.79	−1.14	69,238.65	−22.4
19-250-F-S	75,309.1	73,610.9	−2.3	73,575.7	−2.4	73,835.5	2	73,544.7	−2.4	72,038.6	−4.54	39,672.94	−47.3
20-250-F-L	136,594	134,889	−1.3	134,186	−1.8	132,870	−2.8	135,622	−0.7	131,012.9	−4.26	54,144.64	−60.3
21-250-P-S	113,886	112,427	−1.3	112,480	−1.3	112,428	−1.3	110,062	−3.5	112,747.3	−1.01	69,574.44	−38.9
22-250-P-L	200,402	200,402	0	198,611	−0.9	195,135	−2.7	196,524	−2	198,241.2	−1.09	112,454.5	−43.8
23-250-R-S	164,070	162,960	−0.7	164,070	0	162,115	−1.2	162,115	−1.2	161,804.7	−1.4	103,456	−36.9
24-250-R-L	274,563	271,185	−1.2	271,193	−1.2	272,957	−0.6	269,790	−1.8	269,734.7	−1.79	148,156.7	−46
Average-250	160,804	159,245.65	−1.13	159,019.28	−1.27	158,223.42	−1.10	174,822.60	−1.93	157,596.57	−2.35	87,909.87	−45.3
25-500-F-S	167,080	161,063	−3.7	161,040	−3.8	159,064	−5	161,375	−3.5	160,592.1	−4.04	59,737.27	−64.2
26-500-F-L	259,319	249,171	−4.1	249,661	−3.9	247,986	−4.6	249,312	−4	249,105.7	−4.1	107,852.1	−58.4
27-500-P-S	206,788	203,832	−1.5	205,480	−0.6	200,764	−3	201,590	−2.6	208,287.7	0.72	105,678.8	−48.8
28-500-P-L	331,291	321,471	−3.1	325,522	−1.8	320,872	−3.2	321,155	−3.2	337,358.9	1.82	229,525.6	−30.7
29-500-R-S	281,711	279,105	−0.9	278,233	−1.3	274,419	−2.7	276,392	−1.9	265,040	−6.29	126,298.6	−55.1
30-500-R-L	417,282	407,689	−2.4	409,755	−1.8	407,849	−2.3	409,262	−2	411,967.6	−1.29	259,683.6	−37.7
Average-500	277,245.2	270,388.50	−2.61	271,615.17	−2.20	268,492.33	−3.47	269,847.67	−2.87	272,058.67	−2.20	165,807.7	−40.1
31-1000-F-S	173,826	154,196	−12.7	151,943.2	−12.2	154,357	−12.6	156,106	−11.4	152,358.7	−11.8	33,184.03	−80.9
32-1000-F-L	247,385	221,958	−11.5	221,322	−11.8	224,999	−9.9	226,913	−9	222,089.1	−10.1	92,970.35	−62.4
33-1000-P-S	464,580	450,349	−3.2	450,337	−3.2	442,311	−5	442,218	−5.1	457,759.4	1.47	272,126.9	−41.4
34-1000-P-L	712,918	693,407	−2.8	672,910	−5.9	684,121	−4.2	683,211	−4.3	681,370.5	−3.26	498,835.3	−30
35-1000-R-S	492,083	463,625	−6.1	472,626	−4.1	465,022	−5.8	469,162	−4.9	461,919.6	−4.90	207,134	−57.9
36-1000-R-L	705,167	673,951	−4.6	669,926	−5.3	676,720	−4.4	684,217	−3.1	676,159.7	−2.90	464,549.3	−34.1
Average-1000	465,993.2	405,958	−6.8	439,844.03	−7.08	441,255	−6.98	443,638	−6.3	441,942.83	−5.2	261,466.65	−43.8

Table 7. Cont.

Dataset	K-Means-GRASP-ABC(C)S2		Modified K-Means-GRASP-ABC(C)S1		Modified K-Means-GRASP-ABC(C)S2		ACO-GRASP-ABC(C)S1		ACO-GRASP-ABC(C)S2		ABC(S1)		ABC(S2)	
	Best Profit	Gap	Best Profit	Gap	Best Profit	Gap	Best Profit	Gap	Best Profit	Gap	Best Profit	Gap	Best Profit	Gap
1-20-F-S	30,427.58	0.3	27,007.58	−10.9	27,007.58	−10.9	25,490.76	−15.9	25,490.76	−15.9	30,554.62	0.78	30,315.52	0
2-20-F-L	37,556.07	−4.5	40,104.23	1.9	39,240.95	−0.2	36,423.3	−7.4	36,322.33	−7.6	40,104.23	1.9	36,234.48	−7.8
3-20-P-S	44,334.2	−16.1	51,515.53	−2.5	44,334.2	−16.1	42,944.78	−18.7	34,107.64	−35.4	53,164.53	0.58	52,565.1	−0.5
4-20-P-L	57,350.35	−1.7	57,350.35	−1.7	57,350.35	−1.7	56,603.63	−3	47,402.91	−18.7	59,140.98	1.32	57,286.89	−1.8
5-20-R-S	34,090.58	−6.5	30,827.18	−15.4	29,843.58	−18.1	31,741.08	−12.9	31,741.08	−12.9	<u>36,469.52</u>	0	<u>36,469.52</u>	0
6-20-R-L	42,169.13	−3.6	44,764.07	2.2	43,780.72	0.04	41,071.62	−6.1	40,600.21	−7.2	44,707.23	2.1	42,873.71	−2
Average-20	40,987.99	−5.8	41,928.16	−3.6	40,259.56	−7.4	39,045.86	−10.2	35,944.16	−17.4	44,023.52	1.16	42,624.2	−2
7-50-F-S	21,603.83	−3.7	21,603.83	−3.7	21,603.83	−3.7	21,281.88	−5.1	25,123.15	11.9	24,975.25	11.2	24,998.45	11.3
8-50-F-L	51,001.04	−12.1	51,001.04	−12.1	50,896.3	−12.3	49,128.45	−15.3	50,061.24	−13.7	53,710.99	−7.4	54,288.08	−6.4
9-50-P-S	54,165.39	−6	54,165.39	−6	53,064.14	−7.9	55,643.25	−3.4	55,643.25	−3.4	<u>57,651.71</u>	0	<u>57,651.71</u>	0
10-50-P-L	119,099.6	−10.2	119,099.6	−10.2	116,868.6	−11.8	100,393.3	−24.3	106,544.8	−19.6	<u>126,360</u>	−4.7	133,743.2	0.82
11-50-R-S	28,907.33	−13.2	28,907.33	−13.2	28,907.33	−13.2	31,251.34	−6.2	29,969.16	−10.1	33,058.21	−0.8	33,159.12	−0.5
12-50-R-L	76,697.65	−3.3	76,697.65	−3.3	76,604.84	−3.4	65,792.21	−17	65,752.8	−17.1	77,467.34	−2.3	76,517.36	−3.5
Average-50	58,579.14	−8.3	58,579.14	−8.3	57,990.84	−9.2	53,915.07	−15.6	55,515.73	−13.1	62,203.92	−2.6	63,392.99	−0.8
13-100-F-S	36,188.32	−18.4	35,937.96	−19	38,323.6	−13.6	33,565.46	−24.4	33,566.06	−24.4	41,812.08	−5.8	41,636.8	−6.2
14-100-F-L	63,488.24	−22.5	60,851.27	−25.7	61,351.99	−25.1	52,328.66	−36.1	49,789.52	−39.2	66,375.92	−19	72,595.41	−11.4
15-100-P-S	60,815.37	−14.9	54,190.18	−24.2	60,554.49	−15.3	60,614.78	−15.2	47,498.28	−33.5	69,275.79	−3.1	71,400.63	−0.1
16-100-P-L	104,786.4	−16.1	102,121.7	−18.3	111,728.2	−10.6	95,836.25	−23.3	101,142.5	−19.1	115,579.2	−7.5	128,473.2	2.7
17-100-R-S	61,647.98	−23.7	61,962.79	−23.3	61,962.79	−23.3	52,689.7	−34.8	52,689.7	−34.8	77,484.06	−4.1	77,484.06	−4.1
18-100-R-L	101,013.6	−23.3	105,175.3	−20.2	110,683.8	−16	95,363.69	−27.6	97,568.94	−25.9	119,554.3	−9.3	126,871.5	−3.7
Average-100	71,323.32	−20.1	70,039.87	−21.5	74,100.81	−17	65,066.42	−27.1	63,709.17	−28.6	81,680.23	−8.5	86,410.27	−3.2
19-250-F-S	35,443.33	−52.9	48,079.56	−36.1	39,848.03	−47	36,309.86	−51.7	36,612.19	−51.3	36,309.86	−51.7	36,612.19	−51.3
20-250-F-L	67,014.68	−50.9	70,325.74	−48.5	87,575.52	−35.8	51,239.48	−62.4	62,831.21	−54	51,239.48	−62.4	62,831.21	−54
21-250-P-S	77,941.82	−31.5	81,510.78	−28.4	92,124.73	−19.1	73,654.78	−35.3	73,143.68	−35.7	73,654.78	−35.3	73,143.68	−35.7
22-250-P-L	135,164.3	−32.5	133,567.4	−33.3	161,377.2	−19.4	95,906.32	−52.1	131,668.8	−34.2	95,906.32	−52.1	131,668.8	−34.2
23-250-R-S	97,231.36	−40.7	113,547.4	−30.7	95,471.16	−41.8	107,628.9	−34.4	101,863.1	−37.9	107,628.9	−34.4	101,863.1	−37.9
24-250-R-L	187,891.9	−31.5	180,807.7	−34.1	180,803	−34.1	146,270.4	−46.7	166,523.5	−39.3	146,270.4	−46.7	166,523.5	−39.3
Average-250	100,114.6	−37.7	104,639.8	−34.9	109,533.3	−31.8	85,168.2	−47	95,440.4	−40.6	85,168.2	−47	95,440.4	−40.6
25-500-F-S	74,692.46	−55.2	51,581.38	−69.1	76,481.48	−54.2	61,718.84	−63	54,247.37	−67.5	78,432.78	−53	86,250.21	−48.3
26-500-F-L	124,478.4	−51.9	106,505.3	−58.9	131,572.4	−49.2	72,388.37	−72	93,308.66	−64	112,172	−56.7	126,203.2	−51.3
27-500-P-S	152,566.6	−26.2	107,797.6	−47.8	150,645.3	−27.1	109,238	−47.1	112,111.9	−45.7	136,168.2	−34.1	148,384.5	−28.2

Table 7. Cont.

Dataset	K-Means-GRASP-ABC(C)S2		Modified K-Means-GRASP-ABC(C)S1		Modified K-Means-GRASP-ABC(C)S2		ACO-GRASP-ABC(C)S1		ACO-GRASP-ABC(C)S2		ABC(S1)		ABC(S2)	
	Best Profit	Gap	Best Profit	Gap	Best Profit	Gap	Best Profit	Gap	Best Profit	Gap	Best Profit	Gap	Best Profit	Gap
28-500-P-L	249,759.7	−24.6	225,812.4	−31.8	247,180.6	−25.3	152,573.5	−53.9	196,969.2	−40.5	201,973.8	−39	225,040.5	−32
29-500-R-S	172,397.2	−38.8	108,998.5	−61.3	175,267.1	−37.7	141,743.7	−49.6	128,570.6	−54.3	178,693.9	−36.5	173,003.3	−38.5
30-500-R-L	275,488.2	−33.9	241,454	−42.1	294,424.7	−29.4	167,819.5	−59.7	214,081.3	−48.6	239,675.7	−42.5	261,517.2	−37.3
Average-500	174,897.1	−36.9	140,358.2	−49.3	179,261.9	−35.3	117,580.3	−57.5	133,214.8	−51.9	157,852.7	−43	170,066.5	−38.6
31-1000-F-S	51,188.29	−70.5	39,530.22	−77.2	51,585.38	−70.3	23,266.5	−86.6	25,563.9	−85.2	33,125.19	−80.9	42,197.08	−75.7
32-1000-F-L	97,093.51	−60.7	92,142.15	−62.7	97,772.35	−60.4	24,110.85	−90.2	47,620.03	−80.7	57,137.24	−76.9	73,126.9	−70.4
33-1000-P-S	293,900.2	−36.7	236,797.1	−49	287,774.9	−38	178,654.3	−61.5	209,372.1	−54.9	240,944.9	−48.1	267,814.2	−42.3
34-1000-P-L	536,199.8	−24.7	492,753.1	−30.8	534,689.9	−24.9	346,244.9	−51.4	381,116.4	−46.5	397,930	−44.1	430,894.4	−39.5
35-1000-R-S	264,295.2	−46.2	192,030.8	−60.9	249,078.6	−49.3	124,538.1	−74.6	181,237	−63.1	212,979.1	−56.7	232,298.7	−52.7
36-1000-R-L	458,126.7	−35	407,703.7	−42.1	463,915.7	−34.2	297,212.8	−57.8	335,935.6	−52.3	364,643.1	−48.2	380,268.9	−46
Average-1000	276,004.15	−40.7	243,492.8	−47.7	280,802.8	−39.7	165,671.2	−64.4	196,807.5	−57.7	217,793.3	−53.2	237,766.7	−48.9

The best-known profit was computed in [9] by running each instance for 20 h, and then the best profit was saved as the best-known profit. As can be seen from Tables 6 and 7, our proposed algorithms perform well in terms of profit for small-size instances, with the ABC(S1) returning six new best solutions and two other solutions that were equal to the best-known solutions. The ABC(S2) also achieved three new best solutions and three solutions that are equal to the best-known solution. Also, the modified K-means-GRASP(C)-ABC(C)S1 achieved two new best solutions. In addition, the K-means-GRASP(C)-ABC(C)S1, K-means-GRASP(C)-ABC(C)S2, modified K-means-GRASP(C)-ABC(C)S2, and ACO-GRASP(C)-ABC(C)S2 each produced one new best solution. Overall, the general performance for all proposed algorithms was acceptable and close to the previous algorithms in terms of average profit on these instances.

For the medium- and large-sized instances, though, our proposed algorithms did not do well compared to previous methods, as evident by the large average gap percentage compared to the best-known results shown in Table 6. There are many possible reasons for this:

- Again, the methods used in the scout phase of the ABC seem to have an effect on the performance of the proposed algorithms. The first is the greedy randomised strategy, which selects a random customer pair, meaning the worst pair (either with low revenue or separated by a great distance) may be selected, resulting in poor solution quality. Moreover, in some cases, the neighbourhood operators cannot sufficiently enhance the quality of the solution because of time violations or load constraints. The second method uses the GRASP, which constructs solutions by randomly selecting from the Restricted Candidate List (RCL), which is half the size of the Candidate Solution Set (CSS). Thus, selecting unvisited customers is not a problem for small-sized instances because the number of customer pairs is small, which means the chance of selecting the best customer pair (i.e., that with the highest IR2 value) is high. In contrast, in medium- and large-sized instances, where the size of the RCL is large, the chance of selecting the best customer pair decreases, which affects the quality of the solutions.
- The number of solutions in a population, the number of iterations, and the limit value play important roles in the algorithms' performance. In our algorithms, we rely on small numbers of both solutions and iterations because a fast processing time is a critical demand for daily m-commerce apps. These decreased iteration numbers and population sizes, however, reduce the chances of achieving satisfactory solution quality.
- All previous algorithms were based on VNS, which has the ability to change the size of the neighbourhood search each time the solution quality is not enhanced. Thus, expanding the neighbourhood structure in a systematic way helps the VNS fully explore the neighbourhood and obtain good solutions. In contrast, the diversification feature in our approach is based either on the greedy feature (inserting the customer pair with the highest IR into the best position) or on a combination of greedy and randomised features (inserting a random customer pair into the best position or inserting a random customer pair from RCL into the best position). However, these strategies seem less successful at diversification than the VNS, which relies on an organised neighbourhood exploration mechanism.

Figure 3 compares our proposed algorithms based on the average of the best solutions for each category of instances (highlighted rows in Table 6). It is clear that the ABC in both versions (S1 and S2) still has the preference over all the other algorithms for the instances comprising 20, 50, 100, 250, and 500 customers. In contrast, both versions of K-means-GRASP(C)-ABC(C) and modified K-means-GRASP(C)-ABC(C) outperform the GRASP(V2)-ABC in both versions (S1 and S2) for the 1000-customer instances. The main reasons for the poor performance of clustering algorithms in small- and medium-sized instances is the distribution of customer pairs in K-means and modified K-means, where the clustering process may result in putting a small number of customers within a cluster, which means that the vehicle still has remaining time to serve, but no more customers exist

within that cluster. Thus, we conclude that the clustering algorithms (except ACO) do well in large-sized instances only.

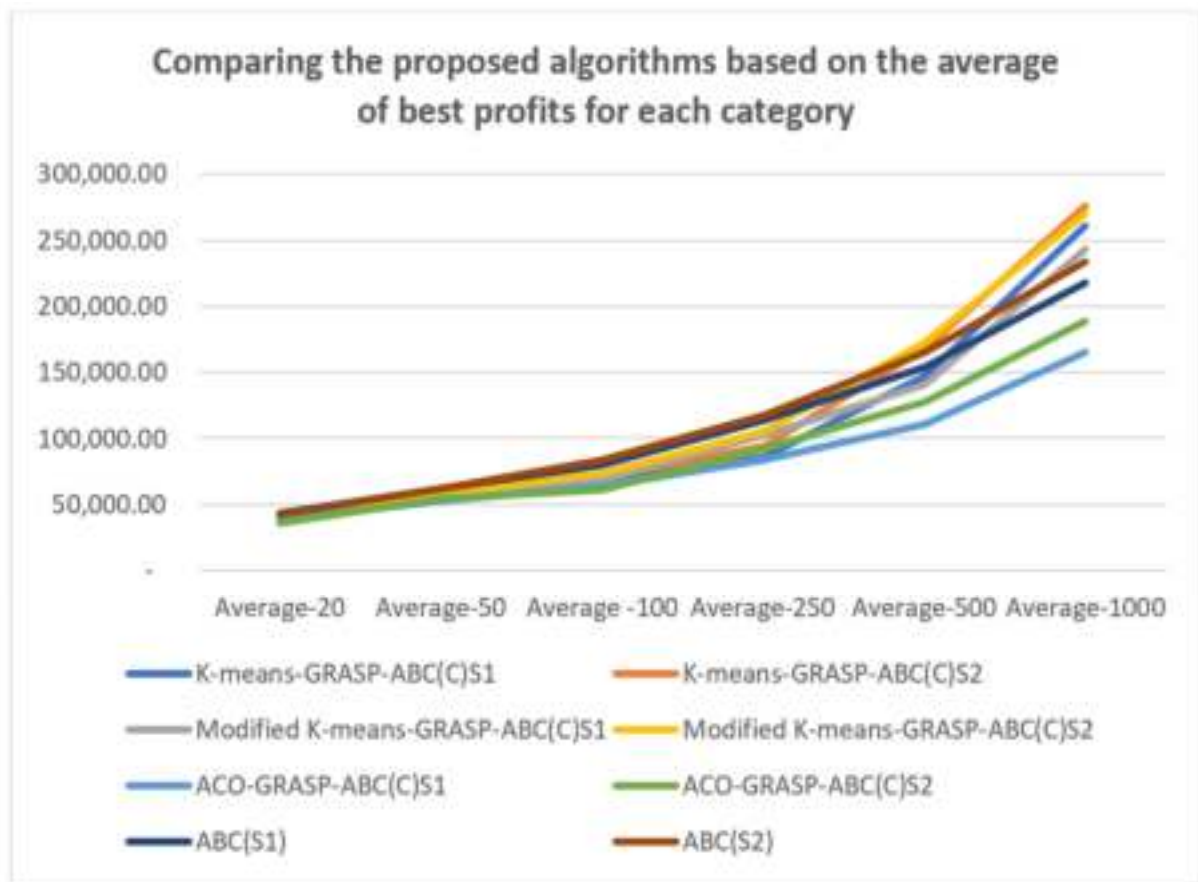


Figure 3. Comparison the proposed algorithms based on the average of best profits for each category.

Regarding the ACO-GRASP(C)-ABC(C)(S1 and S2), it is clear that it has the worst performance among all algorithms. Again, the clustering method of ACO has a significant effect on results. Specifically, the randomness feature of ACO contributes to an inaccurate clustering of customer pairs.

In addition, as seen in the highlighted rows in Table 7, we noticed that the second strategy for scout bees outperforms the first one in many instances because the first strategy includes selecting either the best customer pairs or random pairs, and the random selection option may affect the quality of the solution, while the second strategy selects customer pairs randomly from the RCL, which contains the best customer pairs in the CSS.

6. Conclusions

In this paper, we present eight new metaheuristics to improve the initial solutions for the multi-vehicle profitable pickup and delivery problem (MVPPDP). We proposed two versions of the ABC (Artificial Bee Colony) to improve the initial solutions: the ABC with GRASP and the ABC (C) with GRASP and clustering, i.e., K-means-GRASP (C), modified K-means-GRASP (C), and ACO-GRASP (C). We also have two separate strategies for the scout bee phase. The first strategy, S1, constructs the solution by randomly selecting one of two strategies: greedy and randomised insertion or just greedy insertion. The second strategy, S2, is based on the GRASP. Thus, we have eight different proposed methods. When compared to the state-of-the-art methods in terms of processing time, all eight methods have preference in all instances (except for the 1000-customer instances). Moreover, all proposed algorithms performed well for small-sized instances in terms of profits by achieving thirteen

new best solutions and five equal solutions to the best-known solutions. However, the algorithms slightly lag behind in medium- and large-sized instances due to the greedy randomised strategy and the GRASP that have been used in the scout bee phase. Moreover, our algorithms prioritise minimal solutions and iterations for rapid processing times in daily m-commerce apps, while reducing iteration counts and population sizes reduces the likelihood of obtaining good solution quality.

For future research, more neighbourhood operators will be carefully investigated in the scout bee phase to be applicable within the constraints of the MVPPDP, thus increasing the chances of producing high-quality solutions. Moreover, we will try other population-based metaheuristics, such as GAs, evolutionary algorithms, or other swarm-based metaheuristics that may contribute to new solutions for the MVPPDP.

Author Contributions: Conceptualization, A.I.A.; methodology, A.I.A.; formal analysis, A.I.A.; data curation, A.I.A.; Writing—original draft, A.I.A.; funding acquisition, A.I.A.; Supervision, M.I.H.; writing—review and editing, M.I.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: <http://prolog.univie.ac.at/research/MVPPDP/instances.zip> (accessed on 17 July 2024).

Acknowledgments: The Researchers would like to thank the Deanship of Graduate Studies and Scientific Research at Qassim University for financial support (QU-APC-2024-9/1).

Conflicts of Interest: The authors declare no conflict of interest.

References

- Curry, D. Food Delivery App Revenue and Usage Statistics. 2024. Available online: <https://www.businessofapps.com/data/food-delivery-app-market/> (accessed on 24 January 2024).
- Khalid, H. Benefits of Food Delivery App for Restaurants and Customers. Available online: <https://enatega.com/benefits-of-food-delivery-app/> (accessed on 1 February 2024).
- Team, D.J. Top Food Delivery Apps in Saudi Arabia 2023. Available online: <https://www.digitalgravity.ae/blog/top-food-delivery-apps-in-saudi-arabia/> (accessed on 2 January 2024).
- Gansterer, M.; Küçüktepe, M.; Hartl, R.F. The multi-vehicle profitable pickup and delivery problem. *OR Spectr.* **2017**, *39*, 303–319. [CrossRef]
- Talbi, E.G. *Metaheuristics: From Design to Implementation*; John Wiley & Sons: Hoboken, NJ, USA, 2009; ISBN 978-0-470-27858-1.
- Alhujaylan, A.I.; Hosny, M.I. A GRASP-based solution construction approach for the multi-vehicle profitable pickup and delivery problem. *Int. J. Adv. Comput. Sci. Appl.* **2019**, *10*, 111–120. [CrossRef]
- Alhujaylan, A.I.; Hosny, M.I. Hybrid Clustering Algorithms with GRASP to Construct an Initial Solution for the MVPPDP. *Comput. Mater. Contin.* **2020**, *62*, 1025–1051. [CrossRef]
- Archetti, C.; Speranza, M.G.; Vigo, D. Vehicle routing problems with profits. In *Vehicle Routing: Problems, Methods, and Applications*, 2nd ed.; SIAM: Philadelphia, PA, USA, 2014; Chapter 10; pp. 273–297. ISBN 978-1-61197-358-7.
- Toth, P.; Vigo, D. *Vehicle Routing: Problems, Methods, and Applications*, 2nd ed.; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 2015; ISBN 1611973589.
- Küçüktepe, M. A General Variable Neighbourhood Search Algorithm for the Multi-Vehicle Profitable Pickup and Delivery Problem. University of Vienna 2014. Available online: https://scholar.google.com/scholar?hl=en&as_sdt=0,5&q=A+General+Variable+Neighbourhood+Search+Algorithm+for+the+Multi-Vehicle+Profitable+Pickup+and+Delivery+Problem.+&btnG= (accessed on 22 June 2024).
- Haddad, M.N. An Efficient Heuristic for One-to-One Pickup and Delivery Problems. Ph.D. Thesis, IC/UFF, Niterói, Brazil, 2017.
- Bruni, M.E.; Toan, D.Q. The multi-vehicle profitable pick up and delivery routing problem with uncertain travel times. *Transp. Res. Procedia* **2021**, *52*, 509–516. [CrossRef]
- Karaboga, D. An Idea based on Honey Bee Swarm for Numerical Optimization. Technical Report-tr06, Erciyes University, Engineering Faculty, Computer Engineering Department. 2005. Available online: https://scholar.google.com/scholar?hl=en&as_sdt=0,5&q=An+idea+based+on+honey+bee+swarm+for+numerical+optimization&btnG= (accessed on 22 June 2024).
- Alazzawi, A.K.; Rais, H.M.; Basri, S.; Alsariera, Y.A. PhABC: A Hybrid Artificial Bee Colony Strategy for Pairwise test suite Generation with Constraints Support. In Proceedings of the 2019 IEEE Student Conference on Research and Development (SCORED), Bandar Seri Iskandar, Malaysia, 15–17 October 2019; pp. 106–111, ISBN 9781728126135.
- Szeto, W.Y.; Wu, Y.; Ho, S.C. An artificial bee colony algorithm for the capacitated vehicle routing problem. *Eur. J. Oper. Res.* **2011**, *215*, 126–135. [CrossRef]

16. Wu, B.; Shi, Z. A clustering algorithm based on swarm intelligence. In Proceedings of the 2001 International Conferences on Info-Tech and Info-Net. Proceedings (Cat. No.01EX479), Beijing, China, 29 October–1 November 2001; pp. 58–66. ISBN 0-7803-7010-4.
17. Yao, B.; Hu, P.; Zhang, M.; Wang, S. Artificial bee colony algorithm with scanning strategy for the periodic vehicle routing problem. *Simulation* **2013**, *89*, 762–770. [[CrossRef](#)]
18. Karaboga, D.; Akay, B. A survey: Algorithms simulating bee swarm intelligence. *Artif. Intell. Rev.* **2009**, *31*, 61–85. [[CrossRef](#)]
19. Creutz, M. Microcanonical monte carlo simulation. *Phys. Rev. Lett.* **1983**, *50*, 1411. [[CrossRef](#)]
20. Bansal, J.C.; Sharma, H.; Jadon, S.S. Artificial bee colony algorithm: A survey. *Int. J. Adv. Intell. Paradig.* **2013**, *5*, 123–159. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.



Article

Preptimize: Automation of Time Series Data Preprocessing and Forecasting

Mehak Usmani ¹, Zulfiqar Ali Memon ¹, Adil Zulfiqar ² and Rizwan Qureshi ^{3,*}

¹ Fast School of Computing, National University of Computer and Emerging Sciences, Karachi 65200, Pakistan; k200984@nu.edu.pk (M.U.); zulfiqar.memon@nu.edu.pk (Z.A.M.)

² Department of Electrical Engineering, National University of Computer and Emerging Sciences, Faisalabad Campus, Faisalabad 38000, Pakistan; adil.zulfiqar@nu.edu.pk

³ Center for Regenerative Medicine and Health, Hong Kong Institute of Science and Innovation, Chinese Academy of Sciences, Science Park, Hong Kong SAR 999077, China

* Correspondence: engr.rizwanqureshi786@gmail.com

Abstract: Time series analysis is pivotal for business and financial decision making, especially with the increasing integration of the Internet of Things (IoT). However, leveraging time series data for forecasting requires extensive preprocessing to address challenges such as missing values, heteroscedasticity, seasonality, outliers, and noise. Different approaches are necessary for univariate and multivariate time series, Gaussian and non-Gaussian time series, and stationary versus non-stationary time series. Handling missing data alone is complex, demanding unique solutions for each type. Extracting statistical features, identifying data quality issues, and selecting appropriate cleaning and forecasting techniques require significant effort, time, and expertise. To streamline this process, we propose an automated strategy called Preptimize, which integrates statistical and machine learning techniques and recommends prediction model blueprints, suggesting the most suitable approaches for a given dataset as an initial step towards further analysis. Preptimize reads a sample from a large dataset and recommends the blueprint model based on optimization, making it easy to use even for non-experts. The results of various experiments indicated that Preptimize either outperformed or had comparable performance to benchmark models across multiple sectors, including stock prices, cryptocurrency, and power consumption prediction. This demonstrates the framework's effectiveness in recommending suitable prediction models for various time series datasets, highlighting its broad applicability across different domains in time series forecasting.



Citation: Usmani, M.; Memon, Z.A.; Zulfiqar, A.; Qureshi, R. Preptimize: Automation of Time Series Data Preprocessing and Forecasting. *Algorithms* **2024**, *17*, 332. <https://doi.org/10.3390/a17080332>

Academic Editor: Vittorio Maniezzo

Received: 18 June 2024

Revised: 22 July 2024

Accepted: 26 July 2024

Published: 1 August 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

Keywords: automation; optimization; time series; univariate and multivariate; statistical techniques; machine learning; preprocessing; forecasting; interpolation; single and multiple imputation; missing data

1. Introduction

Time series data, characterized by sequential data points recorded at specific timestamps, present unique challenges in data science, demanding specialized preprocessing techniques to ensure data quality. Time series analysis encompasses data collection, cleaning, preprocessing, and forecasting, tailored to different types of time series, whether numeric or textual, univariate or multivariate, seasonal or nonseasonal. Existing research predominantly focuses on univariate time series, while multivariate time series pose additional complexities such as noise, missing values, outliers, and seasonality. To address these challenges, an automated preprocessing framework integrating various techniques and machine learning algorithms is proposed. Time series analysis, crucial in IoT and financial domains, requires extensive data cleaning and preprocessing due to its distinctive characteristics and challenges. Understanding the properties of the data, such as stationarity and distribution, is essential for selecting appropriate preprocessing techniques. Additionally, deep learning algorithms like GANs and auto-encoders show promising results in preprocessing and forecasting time series data.

The quest for the most accurate forecasting model with minimal error rates in numerical time series analysis requires exhaustive experimentation with various techniques and their combinations. The type of technique applicable to a particular type of data should also be known [1]. However, this process is time-consuming and labor-intensive. Different imputation methods are employed based on the time series properties, such as distribution and seasonality, to handle missing values effectively. With the exponential growth of time series data generated by diverse sources like stock exchanges [2], energy consumption [3], health monitoring systems, and weather recording systems, analysts face increasing workloads. Time series, characterized by sequential data recorded over time intervals, present unique challenges such as handling noise, seasonality, and missing values, which necessitate specialized preprocessing techniques to ensure data quality and improve forecasting accuracy.

Preprocessing, a crucial step in data analysis and machine learning, involves a series of steps to enhance data quality and simplify model training. However, preprocessing is time-intensive, lacks a predefined sequence, and varies across different domains. In the context of time series analysis, preprocessing becomes more complex due to the real-time nature and sequential dependencies of the data. Challenges include handling missing, duplicate, and incorrect data, including outliers. The existing literature predominantly focuses on handling missing data and outlier detection in time series preprocessing.

Outliers, which deviate significantly from the majority of data points, can be univariate or multivariate, with the latter posing greater complexity. Multivariate outlier detection often requires machine learning approaches due to the multidimensional feature space involved. Outlier detection techniques encompass statistical and machine learning methods. Statistical techniques include exponential smoothing, z-score, linear regression models, and interquartile range, while machine learning techniques comprise supervised methods like neural networks and decision trees, and unsupervised methods like cluster analysis. Distance measuring techniques, information theory-based methods, and algorithms like KNN and COF are also utilized.

Handling missing values in time series data involves either removing incomplete records or imputing missing values. Imputation techniques for time series data, including spline interpolation and moving averages, differ from those used in non-time series data due to temporal properties. Traditional methods may not suffice, necessitating advanced techniques like interpolation, auto-encoders, and generative adversarial networks (GANs) for imputation and outlier detection.

Time series data analysis considers properties such as seasonality and trend, categorizing time series into types based on their characteristics. Challenges in time series data extend beyond outliers and missing values to include concept drift, class imbalance, handling trend, and seasonality and data consistency issues.

According to [4], time series analysis is mostly performed using statistical methods; however, machine learning techniques have increasingly gained prominence in time series analysis, promising more accurate results with adequate preprocessing. However, preprocessing remains vital for enhancing data quality and ensuring reliable predictions [5]. Techniques like SCREEN [6] offer constraint-based cleaning methods for time series data, improving accuracy and efficiency in data analysis. Research studies have explored various approaches to enhance preprocessing techniques for time series data. One study [7] utilized meta-learning to select preprocessing alternatives, focusing on transformation effects on algorithm performance across datasets such as Automobile, Credit, Iris, and Vote. While regression trees showed limited effectiveness as a meta-learner, transformations significantly improved outcomes. Another study [8] introduced an iterative minimum repairing and parameter prediction method for anomaly repair, achieving a substantial enhancement in execution speed across datasets including GPS, Intel Lab Data, and China Meteorological data. In a separate investigation [9], Landsat time series imagery combined with trajectory analysis was employed to detect forest instabilities in Myanmar. This study highlighted the correlation between areas of instabilities and harvested trees, indicating

potential applications for selective logging. Change detection in time series was comprehensively analyzed [10], covering preprocessing techniques like atmospheric correction and cloud detection, along with popular automated preprocessing algorithms like LEDAPS and Fmask. Augmenting fully convolutional networks with LSTM-RNN was proposed [11] for time series classification, demonstrating improved performance with attention mechanisms. Additionally, studies [12] addressed data profiling issues, proposing visual-based software solutions for data quality checks and outlier detection techniques using methods like the k-sliding window technique and neural networks [13].

A review [14] highlighted the need for further research in time series cleaning tools and systems, emphasizing the importance of preprocessing for maximizing data potential. Potential areas for future work include error data type analysis, multidimensional time series cleaning algorithms, anomaly detection, and specific application data cleaning techniques. Several innovative systems have been proposed to address challenges in cleaning time series data across various domains. Cleanits [15], for instance, focuses on industrial time series data cleaning, utilizing Bollinger bands for anomaly detection and LSTM for repairing errors. While primarily designed for industrial data, there is potential for its application in other domains, requiring flexibility enhancements.

In the energy sector, knowledge extraction systems have been proposed [16] using clustering techniques, with k-Medoids showing promising results on energy time series. Medical science also benefits from time series data analysis, prompting the development of systems like Clairvoyance [17], facilitating rapid prototyping and optimization of clinical decision systems. Hybrid models and systems like AutoMTS have been introduced for cleaning multivariate time series data [18,19], demonstrating their significance in improving data quality through imputation and outlier detection. AutoML and frameworks like DAEMON [20,21] focus on automating machine learning model fitting and outlier detection in time series data, achieving high accuracy and demonstrating potential for online anomaly detection services [22].

Time series data with seasonality pose unique challenges, such as multi-seasonality and heteroscedasticity, which complicate analysis and prediction. Research [23,24] has addressed these challenges by preprocessing data, building models, and adapting them to original measurements, followed by anomaly detection. Further studies aim to enhance local trend prediction and outlier correction in time series analysis. Non-stationarity has been addressed in studies focusing on volatile and non-volatile data cleaning [25]. Techniques like k-nearest neighbors and sliding windows have been explored [26], with sliding windows performing better, particularly for volatile data like stock market time series [27]. Efforts have been made to measure data quality using indices like the Attribute Value Quality Index (AVQI) and Structured Data Value Quality Index (SDVQI), enabling comparison and improvement of data quality measures.

Optimization algorithms like genetic algorithms (GAs) have been employed to automate feature selection in preprocessing, improving regression models' efficiency [28]. While existing solutions like SCREEN, CLEANITS, DAEMON, and Clairvoyance address specific preprocessing challenges, there is a need for a generic approach spanning multiple domains. Preptimize stands out by offering an integrated solution for preprocessing and model recommendation, making it accessible to non-experts and capable of handling diverse time series challenges. SCREEN and DAEMON focus on anomaly detection and cleaning, while CLEANITS specializes in imputation. Clairvoyance is primarily a forecasting tool utilizing advanced machine learning techniques. Preptimize's broad applicability and user-friendly approach differentiate it from these specialized tools. Various open-source libraries and proprietary SaaS products automate machine learning pipeline generation and optimization, streamlining the preprocessing process [29,30].

Time series values to be forecasted can be nominal, categorical, textual, and numerical. Numerical forecasting is a regression problem which can be solved using various techniques like moving averages, autoregression, and exponential smoothing, as well as advanced methods like ARIMA, SARIMA, VAR, and LSTM [31]. Hybrid approaches

combining statistical and machine learning models are also common [32–34]. However, no single technique suits all time series due to their varying properties, making the selection process time-consuming and complex. To streamline the process of analyzing a time series, a standard procedure should be adopted to automate profiling, preprocessing, optimized model forecasting, evaluation, and selection of the best-performing model with minimal error.

The primary contribution of this work is the introduction of Preptimize, an automated strategy for analyzing time series data that integrates statistical and machine learning techniques to recommend optimal models for analysis. By reading a sample from a large dataset, Preptimize suggests blueprint models based on optimization, including the best combination of preprocessing and forecasting techniques, thereby enhancing forecasting accuracy across various types of time series data. Preptimize streamlines the process of analyzing both univariate and multivariate time series with missing values. On basis of the complexity involved in time series analysis, this research highlights the importance of choosing the right forecasting technique and preprocessing strategy based on the type of time series data to expedite the process of analyzing the time series data. Additionally, this work demonstrated the framework’s potential for broad applicability in time series forecasting, supporting its effectiveness in recommending suitable prediction models.

2. Materials and Methods

Based on the discussion in the previous section, it can be determined that a comprehensive framework is essential that can define standard steps during preprocessing and forecasting time series. The subsequent section has a thorough explanation of the proposed framework called Preptimize. The focus of Preptimize is to automate the process of time series analysis by offering the first blueprint model for further analysis. The proposed framework was implemented using the Python programming language.

Preptimize

Preptimize begins by reading time series data and generating a comprehensive profile that includes distribution analysis, seasonality, and trend detection. It then preprocesses the data by addressing data quality issues such as missing values and outliers using various techniques and evaluates the preprocessed data using multiple forecasting models to identify the best-performing model. The steps are also mentioned in Figure 1. Since the framework design is a big task, this paper focuses on numerical data and missing data problems. This work aims to explain most solutions available to handle missing data in time series and gives the best solution possible. The proposed framework is tested on different datasets including cryptocurrency data, stock market data, and an IoT network’s generated data.

There are numerous techniques and algorithms utilized in the implementation of Preptimize, as outlined in Table 1. The rules that Preptimize follows to recommend potential prediction model blueprints are derived from the existing literature. Therefore, organizing and streamlining these rules and findings is a significant contribution of this work. The basic pseudocode of the proposed framework is given in Figure 2. The following section highlights only the most common techniques employed at various stages of the process.

Table 1. Details of techniques used in the implementation of Preptimize.

Stage	Substage	Details of Techniques Used
Profiling	Statistics	Variance, standard deviation, minimum, maximum, median, mode, missing percentage, quartiles, stationarity, distribution, kurtosis, skewness, cardinality, correlation
	Visualization	Distribution graph, scatter plot, line plot, box plot, histogram, trend and season decomposition graph, correlation heatmap, rolling statistics plot, ACF, PACF

Table 1. Cont.

Stage	Substage	Details of Techniques Used
Preprocessing Techniques	Transformation	Transformation for stationarity using difference, log difference, exponential, additive decompose, multiplicative decompose, rolling mean and standard deviation, ADF statistics
	Missingness Deletion	Transformation for normal distribution using Box–Cox MCAR, MAR, or MNAR
	Univariate Imputation	List-wise, pairwise Moving averages including SMA, exponential, CMA Forward filling (LOCF), backward filling (NOCB) GAIN (based on generative adversarial networks)
	Interpolation	Linear, akima, barycentric, cubic, cubic spline, from derivatives, index, Krogh, nearest, pad, pchip, polynomial, piecewise polynomial, quadratic, slinear, spline, time, zero
	Multivariate Imputation	DataWig, Bayesian Ridge, random forest, K-neighbors, decision tree, Extra Trees, KNNImputer, HotDeck
Forecasting	Univariate	Statistical techniques including moving averages (SMA, exponential, CMA) and other seasonal and nonseasonal techniques including AR, ARIMA, SES, SARIMA, HWES MLP, SVR-RBF, SVR-POLY, decision tree, random forest, RNN, GRU, GRU-bidirectional, Vanilla LSTM, Stacked LSTM, Bidirectional LSTM, CNN-LSTM
	Multivariate	Statistical techniques including VAR, VMA, VARMA MLP, decision tree, random forest, RNN, GRU, GRU-bidirectional, Stacked LSTM

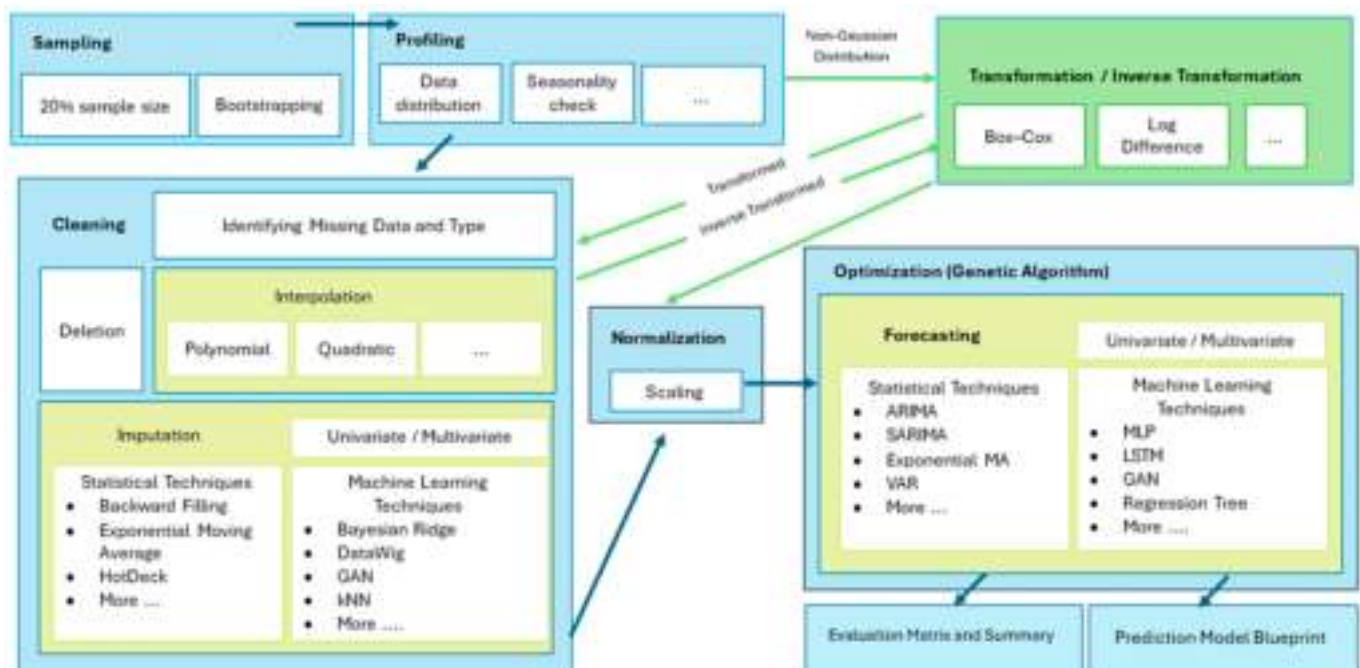


Figure 1. Preptimize: Visual depiction of the systematic approach integrating preprocessing, model selection, and thorough evaluation stages, enhancing efficiency in time series analysis.

```

Initialize dataset
Backup the dataset to handle null values later
# Profile the dataset
Initialize Profile as dictionary
If dataset has multiple columns
    Set Profile as Multivariate
    Print pairwise correlation of dataset
    Display heatmap of correlations
Else
    Set Profile as Univariate
Calculate total records and type of time series (Univariate/Multivariate)
# Prepare profile of each column
For each column in dataset
    Initialize dataAttribute as the column data
    If Univariate, set UnivariateColumnName
    Check for missing data in column
    Calculate nonNullValues and nullValues
    If missing data exist
        Calculate missingPercentage
        Drop null values for further analysis
    Calculate skewness and kurtosis of dataAttribute
    Determine distribution type (Normal/Not Normal)
    Display profile of column with calculated statistics
    Plot seasonal decomposition graph
    Plot basic profile graphs
    Save profile for column in Profile
# Restore original dataset from backup
Initialize missingData and missingColCorrelationCompleteCases
Determine missingness type and correlation
Initialize processedDatasets
# Handle Univariate time series
If Univariate
    Retrieve profile of Univariate column
    If missing data exist
        If missing type is MCAR and missing percentage <= 10%
            Apply list-wise deletion
        Else
            If data are numeric and not skewed
                Apply imputation techniques
            Else if data are not normally distributed
                Apply Box–Cox transformation

```

Figure 2. Cont.

```

        If transformation successful
            Add missing values using multiple techniques and save each dataset in processedDatasets
            Inverse transform data using Box–Cox
        Else
            Set processedDatasets as original dataset
# Forecasting Univariate Time Series
Initialize evaluation_matrix
Apply Genetic Algorithm to select the best prediction model based on highest fitness or lowest RMSE
    Generate different potential solutions including preprocessed dataset, univariate forecasting
    techniques, and parameter settings
    Select the best solution from population using tournament selection
    If missing values remain in preprocessed dataset, drop rows with nulls
    Perform forecasting based on selected forecasting technique and parameters
    Save forecasted data
    Evaluate model by calculating error metrics and update evaluation matrix
Continue until n generations
Show top 5 best performing prediction models' blueprint
# Handle Multivariate time series
Else If Multivariate
    If missing data exist
        Calculate missingDatasetPerc
        If missingDatasetPerc <= 10%
            Apply list-wise deletion
        Else
            Check normality for all columns
            If all columns are normally distributed
                Apply pair-wise deletion
            If columns still have missing data
                Apply single and multiple imputation
            Else
                Apply Box–Cox transformation for non-normal columns
                Apply single and multiple imputation
                Inverse transform data using Box–Cox
    Else
        Set processedDatasets as original dataset
# Forecasting Multivariate Time Series
Initialize evaluation_matrix
Apply Genetic Algorithm to select the best prediction model based on highest fitness or lowest RMSE
    Generate different potential solutions including preprocessed dataset, multivariate forecasting
    techniques, and parameter settings
    Select the best solution from population using tournament selection

```

Figure 2. Cont.

```

    If missing values remain in preprocessed dataset, drop rows with nulls
    Perform forecasting based on selected forecasting technique and parameters
    Save forecasted data
    Evaluate model by calculating error metrics and update evaluation matrix
Continue until n generations
Show top 5 best performing prediction models' blueprint
# Display the findings
Display evaluation matrix and summary of all techniques applied

```

Figure 2. Preptimize: Sequential workflow for time series data preprocessing and model selection. Here, # represents the comment.

2.1. Data Sampling

In the first stage, Preptimize generates a sample from the data to represent the original behavior while reducing the size to 20%. Bootstrapping was applied, and the index was maintained.

2.2. Data Profiling

In this stage, different properties of data are computed, and a profile is created for the attribute. These properties include mean, median, minimum and maximum value, variance, and standard deviation of the data. Data are further explored by finding the distribution type, skewness, and kurtosis of the data. Stationarity of data is also computed, which includes seasonality and trend. A brief description of these techniques is given below.

2.2.1. Data Distribution

Data distribution specifies all likely values for a variable and also quantifies the relative rate of occurrence indicating how frequently a value is appearing. There are different distributions possible, which can be normal, binomial, Poisson, or Bernoulli distribution. Most of the situations follow the normal distribution, which is expressed as a bell-shaped and symmetrical curve around the line $x = \mu$. It indicates an equal number of values are present on both sides of the center. The Gaussian distribution with a mean of 0 and a standard deviation of 1 is the standard Gaussian distribution.

The two properties linked with distribution are skewness and kurtosis. Skewness is a measure of regularity. The series of data is symmetric if it appears the same on both sides of the midpoint. Kurtosis measures the rate of occurrence of outliers at the tail. Data with heavy tails indicate more outliers, and the value of kurtosis is high. The kurtosis of a normal distribution is 3. If less than 3, there are few outliers, and if greater than 3, there are more outliers. Skewness can be negative, which specifies that the tail is on the left side; otherwise, the tail is on the right. If skewness is zero, then the distribution is perfectly symmetrical. The distribution can be considered normal if the values of skewness fall between -3 and $+3$, and kurtosis is in a range of -10 to $+10$ when utilizing the standard error of measurement [35].

2.2.2. Stationarity

A time series whose traits are not subject to time is stationary. Thus, the presence of a trend or seasonality indicates the series is not stationary. The observed property at different times gets affected by trend and seasonality. A time series has three components, which are season, trend, and noise. The season can be defined as the reiterating short-term cycle in the series. The trend is the slope in the data, which can rise or decline with time, whereas the noise is the random change in the series. It is often considered challenging to deal with time series data that contain seasons or trends. Plotting the time series gives instant information about the presence of a season and a trend.

There are two common approaches to deal with seasonal time series. One method is to split the series by removing the seasonal and trend component and then use the series for analysis or use any other transformation technique to obtain the nonseasonal representation of the series. The other way is to use specially designed forecasting techniques for seasonal series like SARIMA.

2.2.3. Identifying the Type of Missing Data

In handling missing data, three categories are identified: missing completely at random (MCAR), missing at random (MAR), and missing not at random (MNAR). In univariate analysis, treating time as a variable adds complexity, requiring consideration of time series traits and covariance for imputation. For MCAR data, univariate imputation is relatively straightforward. However, in multivariate analysis, the association between columns is taken into account. MCAR data are randomly missing and can often be replaced with mean or median values, as the missing data are considered recoverable. MCAR is given by (1), where R is missing data and Y is the complete data.

$$P(R_i|Y_i) = P(R_i) \quad (1)$$

MCAR data lack significant association according to measures like Chi-Square, Cramer's rule, or the correlation coefficient, and typically contain fewer null values. For large datasets, instances with missing data may be omitted or imputed. Deletion methods for MCAR include list-wise deletion, removing records with any null value, suitable when missing data are minimal, and pairwise deletion, retaining data for pairs of quantities with available data points, preserving more data. Imputation becomes necessary when deletion methods are unsuitable. Imputation involves substituting missing data with expected values, with various techniques applied depending on the nature of the data. In the case of MAR, missing data may follow a pattern or sequence, occasionally appearing outside the sequence. When a relationship between missing and existing data is identified, it is considered MAR. Imputation techniques such as mean, median, mode, or interpolation can be employed to handle MAR, with multiple imputation techniques available for non-time series data and interpolation for time series data. MAR is given by (2)

$$P(R_i | Y_i) = P(R_i | Y_{i,o}) \quad (2)$$

In MNAR, specific instances share missing data, making prediction challenging as it relies on unobserved information. If missingness in one column shows little or no correlation with other columns and no strong associations are found upon investigation, and the frequency of missing values is high, this is considered MNAR. MNAR is non-ignorable, with no straightforward method for handling it other than obtaining the missing data. MNAR typically has more null values compared to MCAR and can only be identified through domain knowledge. MNAR is given by (3):

$$P(R_i | Y_i) \neq P(R_i | Y_{i,o}) \quad (3)$$

2.3. Data Visualization

Utilizing various graphs and plots is a valuable approach in time series analysis for understanding data behavior and properties. Preptimize generates a range of visualizations, such as scatter plots, line plots, and decomposition graphs, to provide a comprehensive overview of the time series data. These visual tools help in understanding data behavior, identifying trends and seasonality, and selecting appropriate preprocessing techniques.

The first set of plots offers an overarching perspective of the data, including scatter plots, line plots, box plots, and histograms. Each attribute in a multivariate time series is plotted individually to capture the data's essence.

Another crucial plot displays the data distribution, facilitating the assessment of normality. This evaluation is essential for determining suitable statistical techniques.

A graph illustrating season and trend decomposition is particularly helpful in identifying seasonal patterns within the time series. This insight guides the application of appropriate techniques tailored to seasonal data, potentially necessitating transformations for improved analysis.

To assess data stationarity, Preptimize generates a graph plotting rolling mean and rolling standard deviation against the original data. Stationarity is indicated if these values remain constant over time [36].

The heatmap visualization depicts correlations between multiple variables, with higher correlation levels represented by distinct colors. This aids in understanding the strength of relationships between variables.

Additional graphs focus on missing data, including a matrix and bar plot, which provide insights into the nature of missing data and their potential relationships with other attributes.

2.4. Data Cleaning

The preprocessing phase in time series analysis involves data cleaning addressing various data quality issues such as missing values, outliers, and noise. Unlike other forms of data analysis, time series preprocessing requires specific techniques tailored to their unique characteristics. Preptimize employs a range of imputation techniques to handle missing data, including and not limited to linear interpolation, spline interpolation, and machine learning methods such as GANs. The choice of imputation method is based on the specific characteristics of the dataset, ensuring the best possible data quality for forecasting. With nearly 25 different interpolation techniques available, the framework aims to explore the most suitable approach for each dataset. Transformation techniques are applied when data deviate from normality or exhibit non-stationarity. The Box–Cox transformation is utilized to address non-normality, while methods such as seasonal decomposition are employed to remove trends and seasonality from non-stationary data.

Machine learning approaches, including generative adversarial networks (GANs), are integrated into the framework to enhance missing data prediction. For multivariate time series, iterative imputation models and DataWig are utilized to address missing data challenges.

To determine the optimal imputation technique, a comprehensive evaluation is conducted, considering forecasting performance and evaluation metrics. Alternatively, a brute force approach can be employed to test all possible imputation techniques.

Overall, the preprocessing phase plays a crucial role in preparing time series data for analysis and forecasting, necessitating specialized techniques tailored to their unique characteristics and challenges.

2.4.1. Forward and Backward Filling

Last observation carried forward (LOCF) is a widely used single imputation technique that imputes the last measured value by forward filling. There must be at least one post-baseline measure in LOCF. Next observation carried backward (NOCB) is an analogous method to LOCF but works in the reverse direction by backward filling; that is, it uses the first recorded value after the unavailable value and carries it backward.

2.4.2. Interpolation

Interpolation techniques offer a means to predict missing data points by estimating values between known data points. Linear interpolation, the simplest method, connects adjacent points with straight lines, assuming that the points on the line represent the missing values. Polynomial interpolation, which is more versatile, uses polynomial functions of varying orders to represent relationships between attributes. Lagrange polynomial interpolation and Newton polynomial interpolation are examples of polynomial interpolation methods. Lagrange interpolation, while versatile, may struggle with larger datasets.

Newton polynomial interpolation utilizes a recursive division process to construct complex polynomials, which can be effective for prediction but may lead to numerical complexities and sensitivity to outliers. Spline interpolation, on the other hand, utilizes piecewise polynomials to fit smoother curves to smaller groups of data points, with cubic spline interpolation often preferred for its smoothness and stability.

Other interpolation methods employed in the framework include piecewise polynomial, Akima, Barycentric, Krogh, and Pchip, each offering different approaches to estimating missing values in time series data.

2.4.3. Moving Average

Moving averages, such as simple moving averages (SMAs), cumulative moving averages (CMAs), and exponential moving averages (EMAs), are commonly used for imputation in time series analysis. SMA calculates the average of a specified number of data points, using a simple summation formula. CMA computes the average for all entries up to the current time, with each value equally weighted. EMA employs exponentially decreasing weighting factors to compute the average, with the value at any time determined by a coefficient and the previous EMA value. These moving averages can aid in data imputation, provided the time series is stationary. Stationarity, indicating consistent behavior over time, can be assessed using statistical tests like the Augmented Dickey–Fuller (ADF) test [37] or by observing rolling statistics, such as rolling mean and standard deviation plots.

2.4.4. Deep Learning Algorithms

Data imputation using deep learning techniques, particularly recurrent neural networks (RNNs) and generative adversarial networks (GANs), has emerged as a promising approach in recent literature for time series analysis. RNNs and their variations excel in modeling sequence data, while GANs are adept at generating and imputing missing values. Preptimize leverages a GAN for both single and multiple imputations. The GAN operates through two models, the generator (G) and discriminator (D), which work adversarially to generate realistic data samples and distinguish between real and fake samples, respectively. Training continues until the discriminator is unable to differentiate between real and generated samples, indicating the generator's ability to produce credible examples.

In GANs, D tries to assign the correct label to the samples with maximum probability, whereas G tries to minimize $\log(1 - D(G(z)))$. According to [38], D and G play the two-player minimax game with value function $V(G, D)$, as shown by (4).

$$\min_G \max_D V(D, G) = E_{x \sim p_{data}(x)} [\log D(x)] + E_{z \sim p_z(z)} [\log(1 - D(G(z)))] \quad (4)$$

In Preptimize, a model presented in [39] is used for imputing missing data using a GAN. The method is called Generative Adversarial Imputation Nets (GAIN). In the method, G observes real data, imputes the missing element, and outputs finalized data. D then takes a finalized datum and tries to discover which elements are observed and which are imputed.

2.4.5. Multiple Imputation

Multiple imputations (MIs) offer a comprehensive approach to handling missing values by utilizing the complete set of features to approximate the missing values. MI generates multiple datasets by imputing missing values, allowing for the replacement of missing values with multiple possible values. These imputed values are estimated from each dataset and then aggregated into a final dataset. MI is applicable in cases where the data are missing completely at random (MCAR), when they are missing at random (MAR), and even when the data are missing not at random (MNAR).

Iterative imputation, on the other hand, involves modeling each feature as a function of the other features in a sequential process. Past imputed values are used to predict succeeding features until all missing values are filled. Preptimize utilizes the IterativeImputer library in

Python, which offers various estimators such as the Bayesian Ridge regressor, random forest regressor, K-neighbors regressor, decision tree regressor, and Extra Trees regressor.

Additionally, the KNNImputer technique, provided by the same Python library, employs the k-nearest neighbors approach to impute missing values by considering values of observations for neighboring data points.

DataWig, another method used for imputation, has shown effectiveness in the literature for imputing missing values. DataWig [40] is a deep learning library that leverages MXNet as a backend to generate predictions for missing data.

Furthermore, the hot deck [41] method, known for its computational efficiency, is utilized for imputation. This method imputes missing values by selecting a randomly similar record within a data matrix and using existing values from the donor object. The hot deck operates on one attribute at a time and serves as a single imputation technique.

2.5. Forecasting

There are various prediction approaches that exist in the literature for time series. Examples encompass exponential smoothing, the autoregressive integrated moving average (ARIMA) with seasonal and nonseasonal variants, multiple deep learning techniques, and many others. There are separate techniques for forecasting a univariate and multivariate time series. Preptimize uses a genetic algorithm [42] to explore the vast space of possible models and configurations to find the most suitable one for a given dataset. Each individual in the population represents a potential solution comprising a preprocessing strategy, a forecasting model, and parameter settings. Each individual is evaluated based on a fitness function, which in this case is the RMSE. The lower the RMSE, the better the model. Individuals are selected based on their fitness scores to be parents for the next generation. Mutation is performed based on the bit-flip scheme; 3 individuals are selected from the population at random using tournament selection. This approach helps in automating the model selection process, potentially saving significant time and effort while achieving optimal or near-optimal forecasting accuracy.

2.5.1. Univariate Forecasting

Univariate forecasting involves predicting future values of a single variable based on historical data from consecutive time intervals. The choice of forecasting technique depends on whether the time series exhibits seasonality and trend components.

Two main types of univariate forecasting methods exist: statistical methods and advanced machine learning-based methods. Statistical methods include autoregression (AR), moving average (MA), autoregressive moving average (ARMA), autoregressive integrated moving average (ARIMA), and simple exponential smoothing. Moving average techniques such as simple moving average, exponential moving average, and cumulative moving average are common.

For time series with seasonality and trend, options include seasonal autoregressive integrated moving average (SARIMA), seasonal autoregressive integrated moving average with exogenous regressors, and Holt–Winters exponential smoothing (HWES), with ARIMA and SARIMA being the most successful.

ARIMA combines autoregression, moving average, and differencing to achieve stationarity. Proper identification of ARIMA parameters (p , d , q) is crucial, typically performed by fitting models with different orders using criteria like the Akaike information criterion (AIC) to select the best order. The AIC measures model performance relative to the number of parameters. The autocorrelation function (ACF), and partial autocorrelation function (PACF) cannot be used to identify reliable values for p and q [43]. Residual analysis is essential in ARIMA modeling to ensure residuals resemble white noise, indicating no autocorrelation. The Ljung–Box test is commonly used to assess residual autocorrelation, where p -values above a threshold suggest no significant autocorrelation.

Time series forecasting has garnered significant attention in machine learning and deep learning. Preptimize also considers these ML-based methods of forecasting. The

literature presents various techniques for this purpose, including Long Short-Term Memory (LSTM), support vector regression (SVR), the multi-layer perceptron (MLP), decision trees, random forest, the recurrent neural network (RNN), and its variant the Gated Recurrent Unit (GRU).

Among these techniques, LSTM stands out, known for its effectiveness in handling sequential data. However, traditional RNNs face challenges like vanishing and exploding gradients, leading to longer training times and lower precision. These issues are addressed by LSTM and GRU, which are more efficient in managing gradient problems.

GRU differs from RNNs in its use of update and reset gates to control information flow and address gradient issues. These gates determine how much preceding information is retained or discarded, enhancing the model's efficiency. Variants of GRU, including bidirectional versions, are utilized in the proposed framework.

LSTM, another variant of RNN, comprises multiple LSTM units, each equipped with cell, input, output, and forget gates. These gates play a crucial role in the LSTM model, facilitating the sequential processing of data and prediction generation.

This framework employs various configurations of neural networks for forecasting, including Vanilla LSTM, Stacked LSTM, Bidirectional LSTM, and CNN-LSTM. Vanilla LSTM consists of one hidden layer and one output layer, while Stacked LSTM involves multiple hidden layers. Bidirectional LSTM processes data in both forward and backward directions and then combines the interpretations. CNN-LSTM integrates a CNN to understand sequence parts and then passes them to LSTM for interpretation, forming a hybrid model.

Comparing GRU and LSTM, no clear winner emerges across all performance parameters. GRU executes faster with fewer parameters and less memory usage, while LSTM demonstrates greater precision, especially on larger datasets.

MLP serves as a foundational artificial neural network, featuring fully connected feedforward layers suitable for classification and regression tasks. It comprises at least an input layer, a hidden layer, and an output layer, utilizing nonlinear activation functions to handle linearly non-separable data.

For regression tasks, support vector regression (SVR) offers flexibility in defining error tolerance, seeking to minimize the L_2 -norm of the coefficient vector while allowing for deviations with slack variables. Decision trees, including regression trees, are employed for forecasting time series, while random forest combines multiple regression trees to enhance prediction accuracy through bagging and feature randomness.

2.5.2. Multivariate Forecasting

In the proposed framework, multivariate time series forecasting utilizes a range of statistical and machine learning methods. While statistical approaches for multivariate time series are limited, the framework incorporates key techniques like vector autoregression (VAR), vector moving average (VMA), and vector autoregression moving average (VARMA).

Unlike univariate techniques such as AR and ARIMA, vector-based methods like VAR, VMA, and VARMA are bidirectional, allowing variables to influence each other. However, these techniques require stationarity, often confirmed using the ADF test. VAR establishes relationships between variables through a system of equations, with lag selection guided by criteria like Akaike information criterion (AIC). VMA operates with a moving average matrix and a k -dimensional vector, while VARMA integrates both autoregressive and moving average orders. For machine learning-based forecasting, the framework employs techniques like MLP, regression tree, random forest, RNN, GRU, and LSTM, each detailed in the preceding section.

3. Experimental Results and Discussion

This study evaluates Preptimize, a framework designed for time series data analysis, across various domains and scenarios including stock market data, cryptocurrency data, and IoT-generated power consumption data. Each experiment involves preprocessing the

data, applying various forecasting techniques, and assessing the performance using metrics such as MAE, MSE, RMSE, and MAPE. The details of datasets used are mentioned in Table 2. The framework employs preprocessing and forecasting techniques tailored to each dataset type, with the selection optimized through a genetic algorithm (GA). Preptimize is designed to leverage the GA for optimizing model selection. To ensure a comprehensive understanding of the framework’s functionality, all possible models were evaluated. The GA was adapted to include all potential models, identifying both the best and worst performers. This was achieved by assigning the highest value to the top-performing models. In practical applications, however, the GA typically recommends the five best-performing models based on the lowest RMSE, balancing efficiency and effectiveness. Evaluation metrics include MAE, MSE, RMSE, and MAPE, with detailed analysis provided in a comprehensive report generated by Preptimize. Experiments are conducted separately for univariate and multivariate datasets, covering the stock market, cryptocurrency, and home-based power consumption data. Each experiment is meticulously described, including dataset details, training/testing procedures, and evaluation outcomes. The details and findings of these experiments are given below.

Table 2. Details of datasets used in experiments.

Name	Description
<i>Experiment 1—Univariate Time Series</i>	
Stock Market	Symbol: MSFT (Microsoft) Columns: Date, Close Record Count: 503 Dates: 2020-08-03 to 2022-08-01 Interval: Daily
Stock Market	Same as 1-A, but with missing data
Cryptocurrency	Cryptocurrency: BTC-USD (Bitcoin) Columns: Date, Close Record Count: 748 Dates: 2020-08-01 to 2022-08-17 Interval: Daily
<i>Experiment 2—Multivariate Time Series</i>	
IoT—Power Consumption	Domain: Home Columns: Date, Global_active_power, Global_reactive_power, Voltage, Global_intensity, Sub_metering_1, Sub_metering_2, Sub_metering_3 Record Count: 10,081 Dates: 2007-04-24 to 2007-04-30 Interval: Minute
Stock Market	Symbol: MSFT (Microsoft) Columns: Date, Open, High, Low, Close Record Count: 503 Dates: 2020-08-03 to 2022-08-01 Interval: Daily

3.1. Experiment 1: Univariate Series

Preptimize underwent initial testing on market data including stock and cryptocurrency, a common type of time series data characterized by the fluctuating prices of listings over time. Experiment 1 is divided into three parts. Part 1 utilized complete stock data to assess Preptimize’s performance on high-quality data. Part 2 introduced intentionally missing data to evaluate the framework’s handling of incomplete datasets. Part 3 employed cryptocurrency data with missing values to further test Preptimize’s capabilities. These experiments aimed to gauge the framework’s effectiveness across various scenarios and data qualities in the domain of financial prediction.

In the first part of the experiment, this study explores stock market data from Yahoo Finance, focusing on the Microsoft Corporation (MSFT), listed on the NASDAQ exchange.

Using the yfinance Python library, market data from August 2020 to August 2022 were collected, specifically examining the closing prices on each trading day, excluding weekends. This constitutes a univariate time series dataset, comprising around 500 records, which were divided into training and test sets. Notably, the dataset was complete and devoid of quality issues, obviating the need for preprocessing in the initial phase of the experiment.

A total of 139 models were employed to forecast the time series, with the generated report providing an extensive overview of the data analysis process. Despite space limitations, the report includes essential features such as basic profile characteristics, rolling mean and standard deviation graphs, scatter plots, box plots, distribution graphs, and trend decomposition graphs. Moreover, the report encompasses training logs, error calculations, and performance evaluations for all models. Figure 3a,b illustrate the distribution of the data, indicating normalcy, and depict the decomposition of trend and seasonality within the time series. Comprehensive analyses, including p -values and Augmented Dickey–Fuller (ADF) statistics, suggest the stationarity of the time series with no discernible seasonality. The model performance summary file had results of about 139 models, out of which a few are mentioned in Table 3, showing only the best 5 and worst 5 performers.

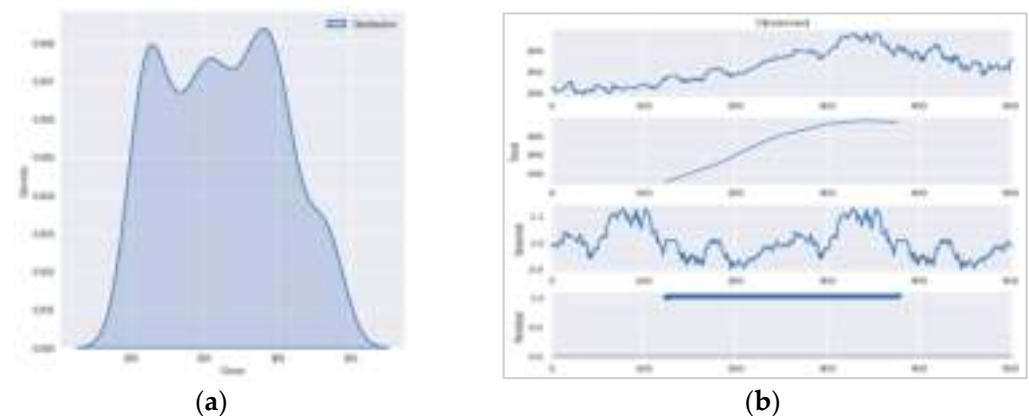


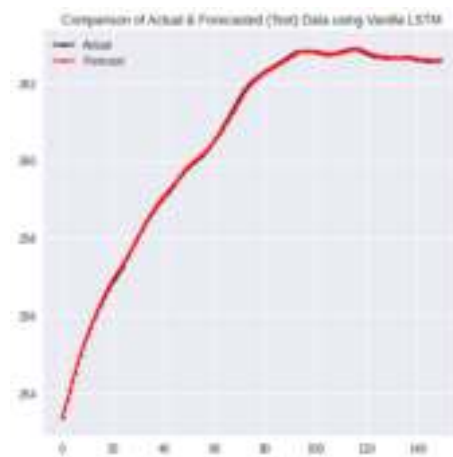
Figure 3. (a) The graph shows normal data distribution of MSFT closing prices for 2 years. (b) Decomposition graph of closing prices to split the trend and seasonality component. The graph represents a clear upward trend and no seasonality.

Various forecasting techniques were applied to a univariate stock market dataset, encompassing both machine learning and statistical methods. Machine learning techniques such as LSTM, GRU, and MLP, along with statistical methods like simple moving average, exponential moving average, and ARIMA, were tested with different hyperparameters including hidden layers, neurons, and epochs. Similarly, various variants of statistical techniques were evaluated, such as different orders for ARIMA. These models underwent evaluation on the test dataset using diverse metrics.

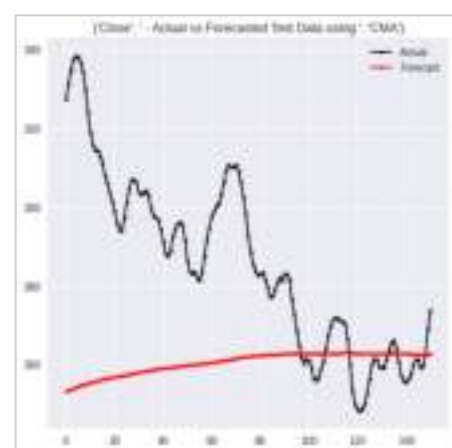
The results indicated that the Vanilla LSTM deep learning model performed best on the dataset, achieving an RMSE of 0.0294 with 50 epochs and 40 units. The predicted values generated by this model were compared with the expected values on the test set, as shown in Figure 4a. Among the statistical techniques, exponential moving average (EMA) performed the best, with an MAE of 2.0925 and an RMSE of 2.5814. EMA was ranked 127th out of 139 models. Conversely, the cumulative moving average (CMA)-based model exhibited the worst performance, with an MAE of 27.2085 and an RMSE of 35.1176. Figure 4b depicts a comparison between the actual values and the forecasted values produced by the CMA model.

Table 3. Model performance summary of experiment based on univariate time series.

Rank	Preprocessing	Model	Settings	MAE	RMSE
<i>Part 1: Stock Market Time Series with No Data Quality Issues</i>					
1	NA	Vanilla LSTM	Epoch = 50, units = 40	0.02	0.03
2	NA	GRU	Epoch = 75, units = 50	0.02	0.03
3	NA	Vanilla LSTM	Epoch = 25, units = 50	0.02	0.03
4	NA	GRU	Epoch = 50, units = 40	0.03	0.04
127	NA	Exponential	-	2.09	2.58
135	NA	Decision tree regressor	Split = MSE	7.57	8.03
136	NA	Random Forest	100 trees, split = MSE	7.69	8.15
138	NA	AR	-	21.11	26.88
139	NA	CMA	-	27.21	35.12
<i>Part 2: Stock Market Time Series with Missing Data</i>					
1	CMA	ARIMA	(7, 1, 2)	0.57	0.61
2	Spline 3	ARIMA	(7, 1, 7)	0.65	0.84
6	Exponential	Exponential	-	1.92	2.43
59	Time	SVR	Kernel = poly	3.64	4.73
64	Derivatives	SVR	Kernel = RBF	3.68	4.75
69	Piecewise polynomial	Stacked LSTM	Epoch = 100, units = 50	3.66	4.76
70	Derivatives	RNN	Epoch = 25, units = 50	3.67	4.76
1269	Polynomial 7	CMA	-	27.82	35.76
1270	Polynomial 9	CMA	-	28.46	36.52
<i>Part 3: Cryptocurrency Time Series with missing data</i>					
1	GAIN	Bidirectional LSTM	Epoch = 100, units = 50	9.09	10.03
2	Index	Stacked LSTM	Epoch = 100, units = 50	8.86	10.64
3	pchip	Vanilla LSTM	Epoch = 100, units = 50	10.82	12.11
10	Exponential	GRU-bidirectional	Epoch = 25, units = 50	12.95	14.67
693	Cubic	AR	-	243.97	291.00
703	GAIN	DT regressor	Split = MSE	260.87	324.39
729	pchip	Random forest	100 trees, split = MSE	292.03	362.15
824	CMA	SVR	Kernel = polynomial	2575.77	2576.76
892	Spline 5	CMA	-	7431.54	9344.14



(a)



(b)

Figure 4. (a) Comparison of actual and forecasted data generated by Vanilla LSTM using 50 epochs and 40 units. (b) Comparison of actual and forecasted data generated by cumulative moving average (CMA).

ARIMA stands out as one of the most effective and efficient techniques for time series forecasting. In Preptimize, ARIMA was implemented with an optimized order of (2, 1, 2), validated through the Ljung–Box test. However, despite these optimizations, the resulting RMSE of 3.6001 was not particularly impressive.

The stock market dataset was used in the second part of the experiment with missing data problems. As mentioned previously, the data were intentionally removed to test the framework performance by comparing them with the original data.

Preoptimize was applied to this dataset containing the missing tuples; the framework generated a report of more than 3500 pages with a detailed description of the steps that were applied to the given dataset, including preprocessing and final analysis. Along with other particulars, the report included details of the missingness type of the data. The framework summarized the data as missing completely at random (MCAR), normally distributed, and stationary, having no season and trend. Figure 5a,b show the trend and season split of the time series and the distribution of the data, respectively.

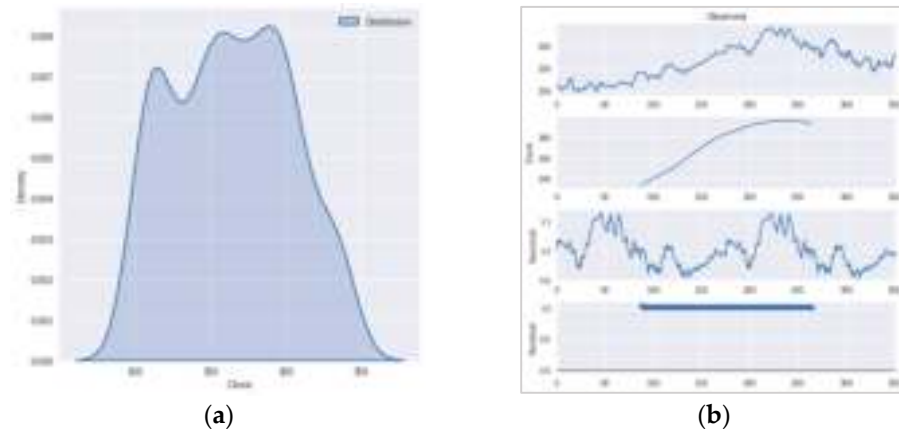


Figure 5. (a) Decomposition graph of closing price to split the trend and seasonality component. The graph shows a slight upward trend and no seasonality. (b) The graph shows normal data distribution of closing price data.

The summary of the performance of all models applied to the data was computed and stored on a sheet. The summary sheet showed there were about 1270 models applied. These 1270 models were generated as a result of different combinations formed by preprocessing and forecasting techniques. For instance, first, a preprocessing strategy like forward fill was applied to impute the missing data, then all other forecasting techniques were applied for univariate forecasting, including statistical and machine learning techniques, and then each model was evaluated using four different metrics of performance evaluation.

The experiment involved trying multiple preprocessing techniques and models, including interpolation, forward and backward filling, moving averages, and deep learning techniques like GAN for imputation. Each model underwent evaluation on the test dataset using various metrics. The results indicated that ARIMA (p, d, q) with order (7, 1, 2) performed best when missing data were imputed using CMA. The Ljung–Box test confirmed the absence of autocorrelation in the white noise and model. This ARIMA model yielded an MAE of 0.56997 and RMSE of 0.6099. The top five models predominantly belonged to ARIMA, utilizing different imputation techniques such as spline order 3, polynomial order 5, backward filling, and padding. After ARIMA, the exponential moving average performed well with multiple imputation techniques including linear, spline, piecewise polynomial, and derivative techniques. The first 85 models, which performed well, were all statistical models. Conversely, the CMA-based model processed by a polynomial with degree 9 performed the worst, with an MAE of 28.4643 and RMSE of 36.5159. Figure 6 illustrates the performance of the best and worst models in the experiment, with Figure 6a displaying the values predicted by the ARIMA (7,1,2) model on the dataset imputed by CMA compared with the original values of the test set, while Figure 6b showcases the performance of the CMA model. Support vector regression with a polynomial kernel model imputed using time interpolation emerged as the best in the machine learning category, yielding an MAE of 3.6430 and RMSE of 4.7306. The errors computed for the mentioned models are tabulated in Table 2.

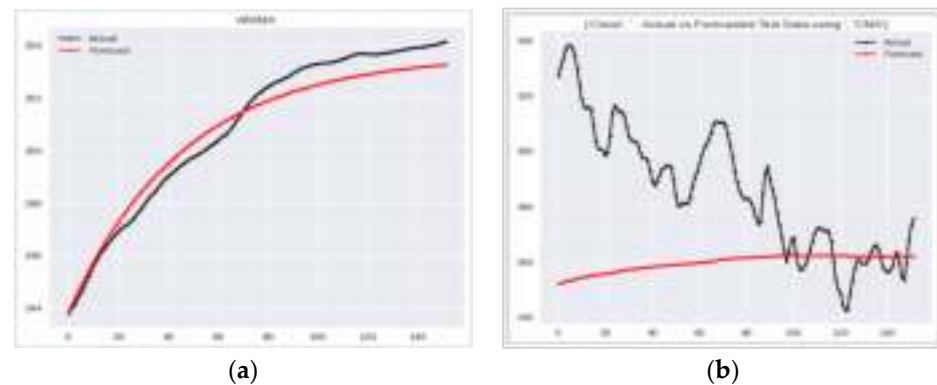


Figure 6. (a) Comparison of actual and forecasted data generated by ARIMA (7,1,2) imputed by CMA. (b) Comparison of actual and forecasted data generated by CMA imputed by polynomial interpolation of degree 9.

The third part of the experiment focused on analyzing cryptocurrency data, particularly Bitcoin (BTC-USD) from the Coinbase exchange. Cryptocurrency operates independently of traditional banking systems, utilizing blockchain technology for transaction authentication and record keeping. The dataset, spanning approximately two years from Aug 2020 to Aug 2022, consisted of 747 records of Bitcoin prices in USD. To evaluate the framework's performance in handling missing data, 30% of the dataset was randomly removed. Preptimize processed the data, identifying them as stationary with no discernible seasonality or trend. Despite the dataset's numeric nature and absence of skewness, it exhibited considerable variance, indicating significant price volatility over the period. Figure 7a,b show the trend and season split of the time series and distribution of the data, respectively. Various imputation techniques, including deep learning methods, interpolation, and moving averages, were employed to address the missing values, with the dataset subsequently split into training and testing sets for forecasting. A total of 892 models were trained and evaluated, with Bidirectional LSTM combined with GAIN imputation emerging as the top performer, achieving an RMSE of 10.028 and MAE of 9.0879. Among statistical techniques, autoregression (AR) showed promise when paired with cubic interpolation, ranking at 693. Conversely, the CMA-based model with spline interpolation and a fifth-degree polynomial exhibited the poorest performance, with an RMSE of 9344.1381 and MAE of 7431.54. Notably, tree-based machine learning algorithms performed suboptimally, with the decision tree regressor and random forest models ranking at 703 and 729, respectively, as shown in Table 2. Performance graphs of the best and worst models are provided in Figure 8.

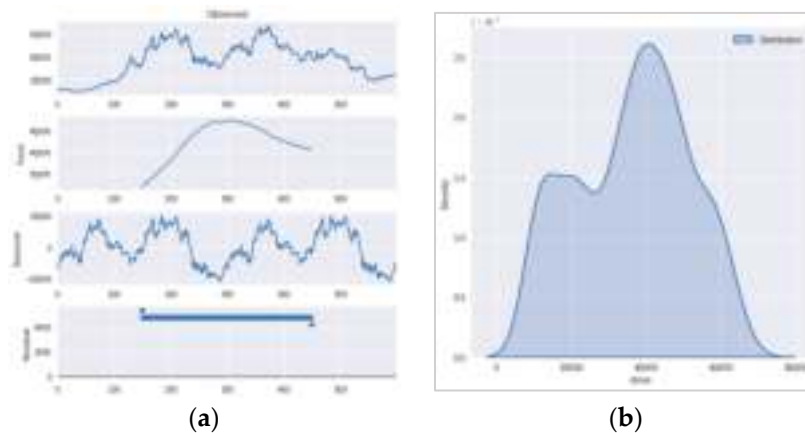


Figure 7. (a) Decomposition graph of closing prices of Bitcoin (b) Data distribution: the graph shows normal data distribution of Bitcoin's closing price in USD.

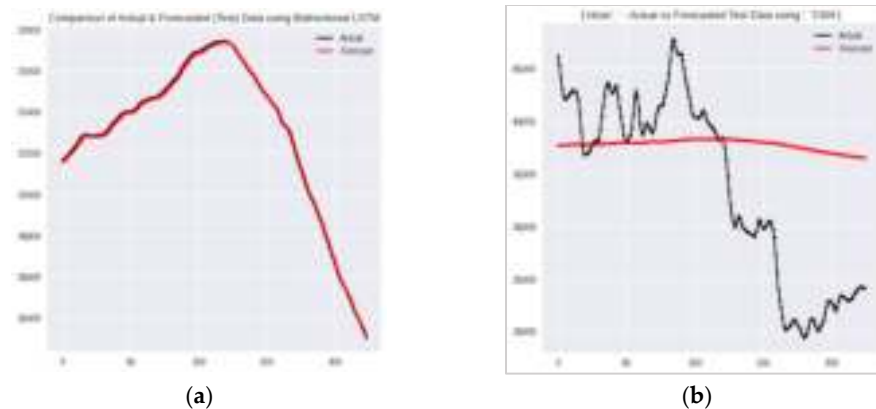


Figure 8. (a) Comparison of actual and forecasted data generated by Bidirectional LSTM imputed by GAIN. (b) Comparison of actual and forecasted series generated by CMA imputed by spline of 5th-degree polynomial.

The results indicate that Preptimize performs well across different types of time series data. For instance, the Vanilla LSTM model showed the best performance on stock market data, with an RMSE of 0.0294, while ARIMA models with various imputation techniques performed best on datasets with missing values, highlighting the framework's versatility and effectiveness.

3.2. Experiment 2: Multivariate Series

The second experiment focused on evaluating Preptimize's performance on multivariate time series data, with and without missing values. The complete data from the stock market were used in the first part, while the second part introduced missing instances using a dataset on individual household electric power consumption.

In the first part, the stock market dataset captured Microsoft's (MSFT) performance based on open, high, low, and close values from August 3, 2020, to August 1, 2022, totaling 503 records. Preptimize began with data profiling, revealing high pairwise correlation among the columns, indicating dependence. However, 'open' and 'high' columns were found to be non-stationary, potentially containing trend and seasonality components, as depicted in Figure 9; the heatmap illustrates the pairwise correlation.

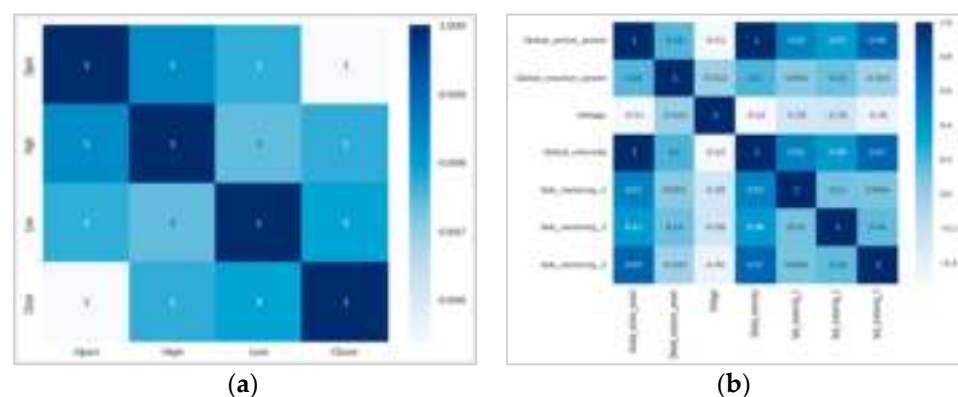


Figure 9. (a) Correlation heatmap showing correlation between pairs in stock market multivariate dataset. (b) Correlation heatmap showing correlation between pairs in power consumption multivariate dataset.

Moving to forecasting, as no preprocessing was needed, Preptimize applied various techniques including VAR, VMA, VARMA, MLP, decision tree regressor, random forest, RNN, GRU, GRU-bidirectional, and Stacked LSTM. Transformation was applied to 'open' and 'high' columns to make them stationary before forecasting. Among the models

mentioned in Table 4, GRU with 50 units and 100 epochs emerged as the best performer, with 3.9242 MAE and 5.1312 RMSE. Figure 10a showcases the performance of the GRU model on the multivariate test set. However, statistical models like VAR, VMA, and VARMA showed poor performance, failing to understand the basic pattern in the multivariate series. Figure 10b showcases the performance of the VAR model on the multivariate data. Furthermore, the individual columns' performance analysis indicated that the 'open' and 'high' columns, initially deemed non-stationary, posed challenges for prediction. Despite applying difference transformation, these columns exhibited predicted values diverging significantly from the original data, leading to high RMSE values.

Table 4. Model performance summary of experiment based on multivariate series.

Rank	Preprocessing	Model	Settings	MAE	RMSE
<i>Part 1: Stock Market Multivariate Time Series with No Data Quality Issues</i>					
1	NA	GRU	Epoch = 100, units = 50	3.92	5.13
2	NA	RNN	Epoch = 100, units = 50	3.93	5.13
3	NA	RNN	Epoch = 25, units = 50	3.95	5.19
4	NA	MLP	Epoch = 100, units = 20	4.04	5.19
17	NA	Random forest	100 trees, split MSE	5.47	7.08
23	NA	VARMA	(2, 1, 3)	407.98	657.08
24	NA	VMA	(0, 1, 1)	365.41	752.85
25	NA	VAR	Order = 5	963.64	1678.42
<i>Part 2: Individual Household Electric Power Consumption Multivariate Time Series with Missing Data</i>					
1	Exponential	GRU	Epoch = 100, units = 20	0.12	0.64
2	CMA	GRU	Epoch = 100, units = 20	0.13	0.64
3	Bayesian Ridge	MLP	Epoch = 100, units = 50	0.13	0.65
4	Random forest	Stacked LSTM	Epoch = 100, units = 50	0.13	0.65
330	KNNImputer 5	VMA	(0,1,1)	0.43	1.88
331	KNNImputer 5	VARMA	(2,1,3)	0.43	1.88
449	polynomial_3	RNN	Epoch = 25, units = 20	9.35	29.31
450	polynomial_3	DT regressor	Split MSE	14.53	58.72

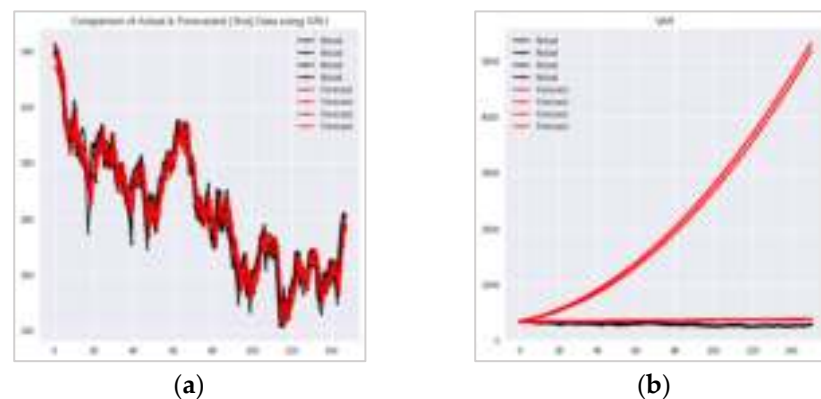


Figure 10. (a) GRU performance comparison for multivariate time series data during testing. (b) VAR model performance comparison for multivariate time series data during testing.

The second part of the multivariate series experiment tested Preptimize using the individual household electric power consumption dataset, aiming to understand its behavior with incomplete multivariate time series data [44]. This dataset comprises seven columns, excluding date and time, recording power consumption over four years with a one-minute sampling rate. With around 2,075,259 instances, the dataset exhibited missing values primarily in April 2007. Details are given in Table 5.

Due to the dataset's size, only one week of data containing missing instances was processed. With 40 preprocessing strategies and 25 forecasting models, a total of 1000 models were evaluated. However, due to computational limitations, the experiment was split into parts, taking several days to generate results. After conducting previous experiments, 18 successful preprocessing techniques were combined with 25 forecasting models for evaluation.

Table 5. Individual household electric power consumption dataset used in Experiment 2.

No.	Column Name	Units	Description
1	global_active_power	Kilowatts	The total active power consumed by the household
2	global_reactive_power	Kilowatts	The total reactive power consumed by the household
3	voltage	Volts	Average voltage
4	global_intensity	Amps	Average current
5	sub_metering_1	Watt-hours	Active energy for kitchen
6	sub_metering_2	Watt-hours	Active energy for laundry
7	sub_metering_3	Watt-hours	Active energy for climate control systems

The comprehensive report generated by Preptimize began by profiling the data, analyzing each attribute individually. A correlation heatmap, depicted previously in Figure 8b, illustrates the pairwise relationships between columns. Notably, global intensity and global active power exhibited a strong positive correlation, while sub-metering 3, associated with climate control systems, correlated strongly with global active power and global intensity. In contrast, global reactive power showed no correlation with other attributes, particularly voltage and sub-metering 3.

Furthermore, the report assessed the missing data percentage, approximately 37%, identifying columns such as sub-metering 1 and sub-metering 2 as non-normally distributed with high kurtosis values. Preptimize classified all four columns with missing data as MNAR type. Attempts to normalize sub-metering 1 and sub-metering 2 distributions were hindered by the possibility of zero or negative values, precluding Box–Cox transformation.

After trend and season decomposition, various imputation techniques, including iterative methods, were applied. Each imputed dataset underwent evaluation using 18 different forecasting models, resulting in a comprehensive performance assessment.

Preptimize generated an evaluation sheet encompassing 450 models, detailing the preprocessing techniques employed during imputation. As shown previously, Table 3 presents the performance metrics of select models, highlighting the best-performing model as GRU imputed with an exponential moving average, yielding an RMSE of 0.6393 and MAE of 0.1225. Conversely, the decision tree regressor, imputed with polynomial interpolation of degree 3, exhibited the poorest performance, with an RMSE of 14.5308 and MAE of 58.7152.

Remarkably, statistical models demonstrated consistent performance when data were imputed using KNNImputer with two neighbors, with VMA, VARMA, and VAR ranking closely. However, some machine learning models fared worse. Performance graphs for GRU and decision tree regressor models are provided in Figure 11a,b, respectively.

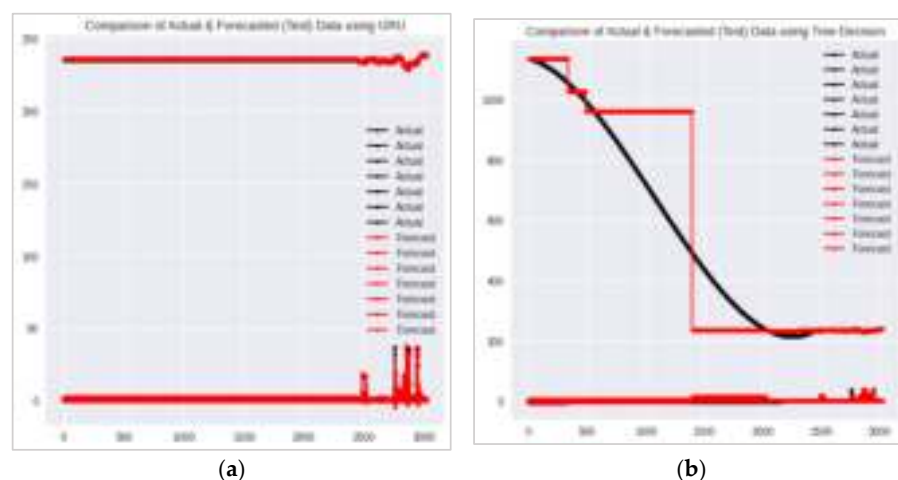


Figure 11. (a) GRU imputed with exponential moving average: performance comparison for power consumption series data during testing. (b) Decision tree imputed with polynomial interpolation of degree 3: performance comparison for power consumption series during testing.

In summary, the experiment revealed exponential moving average, Bayesian Ridge, KNN-based techniques, and DataWig as effective imputation methods, while polynomial interpolation of degree 3, spline interpolation with degree 3, and HotDeck yielded suboptimal results.

4. Conclusions and Discussion

Time series analysis has grown increasingly intricate due to the multitude of techniques available. Choosing the appropriate forecasting method and preprocessing strategy based on the characteristics of the time series is crucial for obtaining reliable results. Proper steps must be followed in time series analysis to identify the best models applicable in practical settings. Statistical and machine learning models, along with various imputation techniques for handling missing data, offer diverse options for analysis. With abundant research detailing each technique's performance on specific datasets, there is a need for a system capable of efficiently applying these techniques to identify the best model through optimization.

Through experiments on various time series datasets, the efficacy of the proposed framework, Preptimize, is evident. Preptimize recommends multiple blueprints for prediction models, serving as a preliminary step towards analyzing the given time series and saving significant time and effort. Researchers can further enhance these models by creating hybrid combinations, tuning parameters, and adding more features to improve forecasting accuracy. Preptimize generates detailed reports covering all data processing steps, along with charts, graphs, statistics, and a comprehensive evaluation sheet showcasing the performance of all models tested on the dataset using different error metrics.

Different experiments were conducted on various datasets to evaluate the performance of this framework. The effectiveness of the framework can be judged by how closely the blueprint model matches those used in the literature and by comparing the accuracy based on RMSE. Benchmark performances for stock market prediction models are based on LSTM, SVR, and hybrid models combining statistical methods with machine learning techniques, often reporting RMSE values as low as 0.02 for stock prices. Preptimize also suggested LSTM and ARIMA for stock price forecasting. Benchmark performances for Microsoft (MSFT) stock prediction reported an RMSE of 0.05 with a hybrid model combining LSTM and RNN [45]. The blueprint model proposed with Preptimize achieved an RMSE of 0.029 using Vanilla LSTM. Bitcoin's high volatility, with significant price fluctuations over short periods, makes it challenging for traditional forecasting models. Bitcoin price prediction reported RMSE values ranging from 4 to 7 for LSTM and random forest models. Preptimize suggested LSTM (RMSE 10.02) for Bitcoin prediction, aligning with findings in the literature. A study [46] reviewed various forecasting models, including feedforward neural networks, LSTM, GRU, ARIMA, and AI techniques, on an individual household electric power consumption dataset to compare their performance. The average RMSE across 12 different models was 0.83, with the lowest RMSE of 0.75 achieved by both LSTM and GRU. In contrast, Preptimize recommended using GRU preprocessed by exponential smoothing, which resulted in an RMSE of 0.639. Based on the discussion, we can conclude that Preptimize effectively recommends suitable prediction models for a given dataset, facilitating rapid decision making with minimal effort.

Preptimize has certain limitations that should be acknowledged. Firstly, it is resource-intensive, requiring significant computational power and time due to its comprehensive approach of applying numerous techniques on a data sample to identify the optimal preprocessing and forecasting methods. Secondly, Preptimize primarily supports datasets with a normal distribution. When dealing with non-normal distributions, it transforms the series to approximate normality, which can add complexity and potential inaccuracies, especially if the transformation does not fully achieve a normal distribution. We aim to address these in future work.

While the proposed framework focuses on addressing the missing data problem in time series analysis, numerous other challenges such as outlier detection, concept drift, and heteroscedasticity persist. Addressing these challenges requires further automation

of time series analysis, a task that will be pursued in future phases of research. Future work could explore extending Preptimize to handle more complex types of time series data, such as those with multiple seasons or irregular intervals. Additionally, integrating more hybrid models based on machine learning techniques and expanding the framework's applicability to other domains like healthcare and environmental monitoring could further enhance its utility.

Furthermore, while we tested the framework's robustness and comprehensiveness against multiple scenarios, we acknowledge that additional scenarios, such as seasonal univariate data, high percentages of missing values, non-normal distributions, multi-seasonal data, and downward trends, could yield further valuable insights. However, including all possible scenarios in the initial work was not feasible. We encourage other researchers to explore this framework with different datasets to address this research gap.

Author Contributions: Conceptualization, methodology, validation, formal analysis, visualization, writing—original draft preparation, M.U.; supervision, Z.A.M.; review and editing, R.Q. and A.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The datasets used in this study are available online. The power consumption dataset is available at <https://archive.ics.uci.edu/ml/datasets/individual+household+electric+power+consumption> (accessed on 1 June 2024). The financial dataset of the stock market is available at <https://aroussi.com/> (accessed on 1 June 2024). The cryptocurrency dataset is available at <https://pypi.org/project/Historic-Crypto/> (accessed on 1 June 2024).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Liu, Z.; Zhu, Z.; Gao, J.; Xu, C. Forecast methods for time series data: A survey. *IEEE Access* **2021**, *9*, 91896–91912. [CrossRef]
2. Asadi, S.; Esmail, H.; Farhad, M.; Masoud, N.M. Hybridization of evolutionary Levenberg–Marquardt neural networks and data pre-processing for stock market prediction. *Knowl. Based Syst.* **2012**, *35*, 245–258. [CrossRef]
3. Di Persio, L.; Fraccarolo, N. Energy consumption forecasts by gradient boosting regression trees. *Mathematics* **2023**, *11*, 1068. [CrossRef]
4. Cryer, J.D.; Kellet, N. *Time Series Analysis: With Applications in R*, 2nd ed.; Springer: New York, NY, USA, 2008; Volume 2, p. 31.
5. Zhou, Q.; Ooka, R. Influence of data preprocessing on neural network performance for reproducing CFD simulations of non-isothermal indoor airflow distribution. *Energy Build.* **2021**, *230*, 110525. [CrossRef]
6. Song, S.; Zhang, A.; Wang, J.; Yu, P.S. SCREEN: Stream data cleaning under-speed constraints. In Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Australia, 31 May–4 June 2015.
7. Bilalli, B.; Abelló, A.; Aluja-Banet, T.; Wrembel, R. Automated data pre-processing via meta-learning. In Proceedings of the International Conference on Model and Data Engineering, Almería, Spain, 21–23 September 2016; Springer: Cham, Switzerland, 2016.
8. Zhang, A.; Song, S.; Wang, J.; Yu, P.S. Time series data cleaning: From anomaly detection to anomaly repairing. *Proc. VLDB Endow.* **2017**, *10*, 1046–1057. [CrossRef]
9. Shimizu, K.; Ponce-Hernandez, R.; Ahmed, O.S.; Ota, T.; Win, Z.C.; Mizoue, N.; Yoshida, S. Using Landsat time series imagery to detect forest disturbance in selectively logged tropical forests in Myanmar. *Can. J. For. Res.* **2017**, *47*, 289–296. [CrossRef]
10. Zhu, Z. Change detection using landsat time series: A review of frequencies, preprocessing, algorithms, and applications. *ISPRS J. Photogramm. Remote Sens.* **2017**, *130*, 370–384. [CrossRef]
11. Karim, F.; Majumdar, S.; Darabi, H.; Chen, S. LSTM fully convolutional networks for time series classification. *IEEE Access* **2017**, *6*, 1662–1669. [CrossRef]
12. Gschwandtner, T.; Erhart, O. Know your enemy: Identifying quality problems of time series data. In Proceedings of the IEEE Pacific Visualization Symposium (PacificVis), Kobe, Japan, 10–13 April 2018.
13. Jeenanunta, C.; Abeyrathna, K.D.; Dilhani, M.S.; Hnin, S.W.; Phyo, P.P. Time series outlier detection for short-term electricity load demand forecasting. *Int. Sci. J. Eng. Technol. (ISJET)* **2018**, *2*, 37–50.
14. Wang, X.; Wang, C. Time series data cleaning: A survey. *IEEE Access* **2019**, *8*, 1866–1881. [CrossRef]
15. Ding, X.; Wang, H.; Su, J.; Li, Z.; Li, J.; Gao, H. Cleanits: A data cleaning system for industrial time series. *Proc. VLDB Endow.* **2019**, *12*, 1786–1789. [CrossRef]

16. Ruiz, L.; Pegalajar, M.; Arcucci, R.; Molina-Solana, M. A time-series clustering methodology for knowledge extraction in energy consumption data. *Expert Syst. Appl.* **2020**, *160*, 113731. [[CrossRef](#)]
17. Jarrett, D.; Yoon, J.; Bica, I.; Qian, Z.; Ercole, A.; Schaar, M.V.D. Clairvoyance: A Pipeline Toolkit for Medical Time Series. In Proceedings of the International Conference on Learning Representations, Virtual, 3–7 May 2021.
18. Desai, V.; Dinesha, H.A. A Hybrid Approach to Data Pre-processing Methods. In Proceedings of the IEEE International Conference for Innovation in Technology (INOCON), Bangalore, India, 6–8 November 2020.
19. Sousa, R.; Amado, C.; Henriques, R. AutoMTS: Fully autonomous processing of multivariate time series data from heterogeneous sensor networks. In Proceedings of the International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness, Virtual, 29–30 November 2020; Springer: Cham, Switzerland, 2020.
20. Chen, X.; Deng, L.; Huang, F.; Zhang, C.; Zhang, Z.; Zhao, Y.; Zheng, K. Daemon: Unsupervised anomaly detection and interpretation for multivariate time series. In Proceedings of the 2021 IEEE 37th International Conference on Data Engineering (ICDE), Chania, Greece, 19–22 April 2021; pp. 2225–2230.
21. Chauhan, K.; Jani, S.; Thakkar, D.; Dave, R.; Bhatia, J.; Tanwar, S.; Obaidat, M.S. Automated machine learning: The new wave of machine learning. In Proceedings of the 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA), Bangalore, India, 5–7 March 2020.
22. Sarafanov, M. AutoML for Time Series: Definitely a Good Idea. Available online: <https://towardsdatascience.com/automl-for-time-series-definitely-a-good-idea-c51d39b2b3f> (accessed on 1 September 2023).
23. Sun, B.; Ma, L.; Shen, T.; Geng, R.; Zhou, Y.; Tian, Y. A Robust Data-Driven Method for Multiseasonality and Heteroscedasticity in Time Series Preprocessing. *Wirel. Commun. Mob. Comput.* **2021**, *2021*, 6692390. [[CrossRef](#)]
24. Zhang, G.P.; Min, Q. Neural network forecasting for seasonal and trend time series. *Eur. J. Oper. Res.* **2005**, *160*, 501–514. [[CrossRef](#)]
25. Ranjan, K.G.; Prusty, B.R.; Jena, D. Comparison of two data cleaning methods as applied to volatile time-series. In Proceedings of the International Conference on Power Electronics Applications and Technology in Present Energy Scenario (PETPES), Mangalore, India, 29–31 August 2019.
26. Ranjan, K.G.; Tripathy, D.S.; Prusty, B.R.; Jena, D. An improved sliding window prediction-based outlier detection and correction for volatile time-series. *Int. J. Numer. Model. Electron. Netw. Devices Fields* **2021**, *34*, e2816. [[CrossRef](#)]
27. Lv, P.; Wu, Q.; Xu, J.; Shu, Y. Stock Index Prediction Based on Time Series Decomposition and Hybrid Model. *Entropy* **2022**, *24*, 146. [[CrossRef](#)]
28. Brunel, B.; Alsamad, F.; Piot, O. Toward automated machine learning in vibrational spectroscopy: Use and settings of genetic algorithms for pre-processing and regression optimization. *Chemom. Intell. Lab. Syst.* **2021**, *219*, 104444. [[CrossRef](#)]
29. Kumar, S. 8 AutoML Libraries to Automate Machine Learning Pipeline. 2020. Available online: <https://medium.com/swlh/8-automl-libraries-to-automate-machine-learning-pipeline-3da0af08f636> (accessed on 1 August 2023).
30. Jang, W.-J.; Lee, S.-T.; Kim, J.-B.; Gim, G.-Y. A study on data profiling: Focusing on attribute value quality index. *Appl. Sci.* **2019**, *9*, 5054. [[CrossRef](#)]
31. Ghaderpour, E.; Pagiatakis, S.D.; Hassan, Q.K. A survey on change detection and time series analysis with applications. *Appl. Sci.* **2021**, *11*, 6141. [[CrossRef](#)]
32. Zou, H.; Yang, Y. Combining time series models for forecasting. *Int. J. Forecast.* **2004**, *20*, 69–84. [[CrossRef](#)]
33. Lim, B.; Zohren, S. Time-series forecasting with deep learning: A survey. *Philos. Trans. R. Soc. A Math. Phys. Eng. Sci.* **2021**, *379*, 20200209. [[CrossRef](#)]
34. Abbasimehr, H.; Paki, R.; Bahrini, A. A novel approach based on combining deep learning models with statistical methods for COVID-19 time series forecasting. *Neural Comput. Appl.* **2022**, *34*, 3135–3149. [[CrossRef](#)] [[PubMed](#)]
35. Brown, T.A. *Confirmatory Factor Analysis for Applied Research*; The Guilford Press: New York, NY, USA, 2006.
36. Haroon, D. Time Series-Differencing. In *Python Machine Learning Case Studies: Five Case Studies for the Data Scientist*; Apress: New York, NY, USA, 2017; p. 110.
37. Agiakloglou, C.; Newbold, P. Empirical evidence on Dickey-Fuller-type tests. *J. Time Ser. Anal.* **1992**, *13*, 471–483. [[CrossRef](#)]
38. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial networks. *Commun. ACM* **2020**, *63*, 139–144. [[CrossRef](#)]
39. Yoon, J.; Jordon, J.; Schaar, M. Gain: Missing data imputation using generative adversarial nets. In Proceedings of the International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018.
40. Biessmann, F.; Rukat, T.; Schmidt, P.; Naidu, P.; Schelter, S.; Taptunov, A.; Lange, D.; Salinas, D. DataWig: Missing Value Imputation for Tables. *J. Mach. Learn. Res.* **2019**, *20*, 1–6.
41. Joenssen, D.W.; Bankhofer, U. Hot deck methods for imputing missing data. In Proceedings of the International Workshop on Machine Learning and Data Mining in Pattern Recognition, Berlin, Germany, 13–20 July 2012.
42. Holland, J.H. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*; MIT Press: Cambridge, MA, USA, 1992.
43. McDowall, D.; McCleary, R.; Bartos, B.J. *Interrupted Time Series Analysis*, 21st ed.; SAGE: New York, NY, USA, 1980; p. 41.
44. Dua, D.; Graff, C. *UCI Machine Learning Repository*; University of California, School of Information and Computer Science: Irvine, CA, USA, 2019. Available online: <http://archive.ics.uci.edu/ml> (accessed on 1 June 2024).

45. Sourav, D.; Apan, P.; Sayan, S.; Sayan, G.; Udatya, D.; Chandra, D.; Shilpi, B. A Novel Hybrid Model Using Lstm and Rnn for Stock Market Prediction. *Int. J. Eng. Res. Technol.* **2024**, *13*. [[CrossRef](#)]
46. Gasparin, A.; Slobodan, L.; Cesare, A. Deep learning for time series forecasting: The electric load case. *CAAI Trans. Intell. Technol.* **2022**, *7*, 1–25. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.

Article

Deep Learning-Based Boolean, Time Series, Error Detection, and Predictive Analysis in Container Crane Operations

Amruta Awasthi , Lenka Krpalkova and Joseph Walsh * 

School of Science, Technology, Engineering and Mathematics (STEM), Munster Technological University (MTU), Kerry Campus, V92 HD4V Kerry, Ireland; amruta.awasthi@research.ittralee.ie (A.A.); lenka.krpalkova@mtu.ie (L.K.)

* Correspondence: joseph.walsh@mtu.ie

Abstract: Deep learning is crucial in marine logistics and container crane error detection, diagnosis, and prediction. A novel deep learning technique using Long Short-Term Memory (LSTM) detected and anticipated errors in a system with imbalanced data. The LSTM model was trained on real operational error data from container cranes. The custom algorithm employs the Synthetic Minority Oversampling TEchnique (SMOTE) to balance the imbalanced data for operational data errors (i.e., too few minority class samples). Python was used to program. Pearson, Spearman, and Kendall correlation matrices and covariance matrices are presented. The model's training and validation loss is shown, and the remaining data are predicted. The test set (30% of actual data) and forecasted data had RMSEs of 0.065. A heatmap of a confusion matrix was created using Matplotlib and Seaborn. Additionally, the error outputs for the time series for the next n seconds were projected, with the n seconds input by the user. Accuracy was 0.996, precision was 1.00, recall was 0.500, and f1 score was 0.667, according to the evaluation criteria that were produced. Experiments demonstrated that the technique is capable of identifying critical elements. Thus, future attempts will improve the model's structure to forecast industrial big data errors. However, the advantage is that it can handle imbalanced data, which is usually what most industries have. With additional data, the model can be further improved.



Citation: Awasthi, A.; Krpalkova, L.; Walsh, J. Deep Learning-Based Boolean, Time Series, Error Detection, and Predictive Analysis in Container Crane Operations. *Algorithms* **2024**, *17*, 333. <https://doi.org/10.3390/a17080333>

Academic Editors: Mateus Mendes and Balduino Mateus

Received: 16 June 2024

Revised: 20 July 2024

Accepted: 24 July 2024

Published: 1 August 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

Keywords: predictive analytics; deep learning; data analytics; time series Boolean data; error detection; deep neural networks; Recurrent Neural Networks (RNNs); Long Short-Term Memory (LSTM) networks; Synthetic Minority Oversampling TEchnique (SMOTE); container cranes; maritime industry

1. Introduction

1.1. Background

The container shipping market is a USD 116.04 billion market in 2024 and is expected to grow to a size of USD 134.03 Billion by 2029 [1]. As the global trade volume escalates, the imperatives for heightened productivity and operational efficacy within port facilities intensify concomitantly. Alongside these demands, ensuring the safety and reliability of container crane operations remain paramount [2]. Container cranes play a pivotal role in modern maritime logistics, facilitating the efficient movement of goods between ships and shores in container terminals worldwide [3]. The towering gantry cranes are indispensable in loading and unloading containers from ships and moving them within the terminal yard.

Container cranes, which operate along the quayside, use complex technology such as spreader beams to transfer containers between ships and terminals, contributing to overall port efficiency. This operation entails precisely arranging the crane to match with various bays on the ship, allowing access to containers located along the vessel's length. Furthermore, in addition to vessel operations, container cranes may be used to stack containers within the terminal yard, allowing for interim storage or onward

transportation [4]. The cranes generate enormous amounts of data that can be analysed, and valuable insights can be obtained for efficient, better, and faster decision-making by industries.

Recently, growing research has focused on data-driven fault diagnostic methods, particularly those based on neural networks. This is due to advancements in sensor technology and increased data storage capacity, which have enhanced the data processing capabilities of neural network-based methods. Failure to identify process flaws can have detrimental effects on productivity, safety, quality of output, and economy of the process. Efficient and advanced automated diagnostic equipment is crucial in modern enterprises to detect, diagnose, and repair abnormal process behaviours. The primary goal of fault detection and isolation is to promptly alert operators to potential issues, enabling them to take necessary measures to avert system failure following the occurrence of problems [5].

Timely error detection and prediction are crucial in optimising crane performance and ensuring safety within container crane operations.

The importance of timely error detection can be understood through several vital perspectives:

1. **Optimisation of Productivity and Efficiency:** Timely error detection and prevention optimise crane performance and operational efficiency. By identifying and addressing errors promptly, operators can minimise the impact on productivity and throughput, ensuring that cranes operate at maximum capacity and efficiency [6]. This can be achieved by predicting errors early so that they do not turn into faults and result in failures. This performance optimisation contributes to container terminal operations' overall productivity and profitability.
2. **Prevention of Operational Disruptions:** Timely error detection allows for the early identification of potential issues or malfunctions in crane operations [7]. By promptly identifying errors, operators can take corrective actions to prevent operational disruptions, such as crane breakdowns or delays in cargo handling. This proactive approach helps to maintain the smooth flow of operations within container terminals, minimising downtime and ensuring the efficient utilisation of resources [8].
3. **Enhancement of Safety Protocols:** Early detection of errors enables operators to implement safety protocols and procedures to mitigate risks and hazards associated with crane operations. By promptly addressing errors, operators can prevent accidents, injuries, and damage to cargo or infrastructure, safeguarding personnel's well-being and preserving the integrity of terminal facilities [9].
4. **Reduction of Maintenance Costs:** Detecting, diagnosing, and predicting errors early can help reduce maintenance costs associated with crane operations. By identifying potential errors before the said errors escalate into significant failures, operators can implement preventive and prescriptive maintenance measures to address underlying issues and prolong the lifespan of equipment [10]. This proactive maintenance approach helps to minimise downtime, repair costs, and operational disruptions, ultimately leading to cost savings for terminal operators [11].
5. **Compliance with Regulatory Standards:** Timely error detection ensures compliance with regulatory standards and industry regulations governing crane operations. By promptly identifying and addressing errors, operators can maintain adherence to safety guidelines and operational protocols mandated by regulatory authorities. This compliance not only ensures the safety of personnel and assets but also helps to avoid potential fines and legal liabilities associated with non-compliance [12,13].

In summary, timely error identification and prediction are essential in optimising crane performance, ensuring safety in container crane operations, and benefiting stakeholders across the maritime supply chain.

1.2. Research Objectives, Motivation, and Contributions

One critical aspect of safeguarding safety, efficiency, and reliability in container terminal operations is detecting and mitigating errors that may occur during crane operations.

Errors in quay container crane operations can have significant implications, ranging from potential damage to cargo and infrastructure to disruptions in supply chain logistics. Moreover, human error, equipment malfunction, and environmental contingencies contribute to the complexity of error detection and pose challenges to ensuring uninterrupted operations [14]. System failures can be prevented by identifying and anticipating errors, resulting in a more robust and durable system. Error prediction in container crane operations involves using data analysis, machine learning, and deep learning methodologies to predict prospective errors or anomalies that may occur during the operation of container cranes [6].

This research aims to develop proactive strategies for error detection and prediction by harnessing the power of deep learning, data analytics, and predictive modelling, thereby enhancing container terminal operations' safety, efficiency, and reliability. Addressing this problem requires a comprehensive understanding of container terminals' operational dynamics and the application of advanced AI technologies and methodologies for error detection, diagnosis, and prediction [15].

By analysing diverse data sources, including crane sensor data, past operation logs, weather conditions, and other pertinent aspects, predictive models can be created to foresee and minimise problems in advance. Implementing this proactive strategy enhances safety, effectiveness, and overall efficiency in container terminal operations.

Research findings in this domain have significant implications for the maritime industry, port authorities, terminal operators, and other stakeholders involved in container terminal operations. The objective is to enhance operational safety, minimise downtime, optimise resource utilisation, and improve overall terminal performance by effectively detecting and mitigating errors in operational data. The scope of this paper lies in exploring how deep learning can enhance error detection, prediction, and time series analysis in container crane operations, which is currently a critical need in the maritime industry and will help to enhance operational efficiency and safety.

The potential of this research extends beyond container cranes to other industries with imbalanced data challenges—where rare events or anomalies occur can benefit significantly. Other industrial sectors can significantly benefit from this research, such as manufacturing, in identifying equipment failures or defects early in production lines; the energy sector, in predicting anomalies in power generation or distribution systems; and transportation, in detecting irregularities in vehicle performance or safety systems. Additionally, organisations can target interventions and preventive measures by identifying critical elements associated with errors.

For instance:

Maintenance: Prioritise maintenance schedules for components most likely to cause errors;

Process Optimisation: Adjust operational procedures based on error patterns;

Resource Allocation: To minimise disruptions, allocate resources (personnel, materials) strategically.

This research is motivated by the urgency of improving error recognition and prediction in container cranes in the multibillion-dollar container shipping industry, which is the backbone of the current worldwide economy. Traditional rule-based techniques frequently fail because they cannot manage complicated, dynamic settings and uneven data distributions. Deep learning, particularly LSTM networks, provides a viable solution to these shortcomings. By capturing temporal connections and learning from previous data, LSTMs can model complex patterns and increase accuracy.

1.3. Deep Learning Overview

Deep learning is a branch of artificial intelligence (AI) that involves training computer systems to perform tasks such as recognising patterns, making decisions, and solving problems without being explicitly programmed to do so. It uses algorithms known as neural networks, which are inspired by the structure and function of the human brain [16]. Data-driven machine health monitoring is becoming increasingly prevalent in modern

production systems because low-cost sensors are widely deployed and connected to the internet. Deep learning offers valuable methods for processing and evaluating large-scale operational data [17]. Deep learning enables computers to recognise patterns, classify information, and make highly accurate predictions by learning from large amounts of data. Deep learning employs neural networks that consist of numerous layers, known as deep architectures, to acquire knowledge from input and generate predictions. This technology has applications in various fields, including image and speech recognition, natural language processing, autonomous vehicles and, in this case, the manufacturing industry.

Several key factors make the utilisation of deep learning paramount in error detection, diagnosis, and prediction in various industries, including quay container crane operations:

1. Deep learning algorithms, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), have demonstrated exceptional capability in handling complex and high-dimensional data [18]. In container crane operations, where data streams from various sensors, control systems, and operational logs are abundant and diverse, deep learning models can effectively process these data to identify subtle patterns and anomalies indicative of errors.
2. Deep learning models excel in capturing temporal dependencies and dynamic relationships within sequential data [19]. In container crane operations, where crane movements, cargo handling processes, and environmental conditions evolve, deep learning algorithms can analyse historical data to predict future states, enabling proactive error detection and prevention [20].
3. The scalability and adaptability of deep learning models make them well-suited for real-time and offline monitoring and decision-making in dynamic operational environments. By continuously learning from new data and adapting to changing conditions, deep learning-based error detection systems can enhance operational safety, optimise resource utilisation, and mitigate potential risks in container crane operations [21,22].

Overall, the importance of utilising deep learning for error detection, diagnosis, and prediction in container crane operations lies in its ability to harness the power of advanced data analytics to enhance operational safety, efficiency, and resilience in maritime logistics. As technology advances, leveraging deep learning algorithms will become increasingly indispensable for ensuring container terminals' seamless and secure operation in the global supply chain ecosystem.

By leveraging advanced deep learning techniques, the research aims to address the inherent complexities of error detection and prediction in container crane operations, where operational data are represented as discrete Boolean values over time. The proposed algorithm seeks to effectively capture temporal dependencies and subtle patterns within Boolean time series data, enabling accurate anticipation of potential errors before occurrence during crane operations. Through empirical validation and performance assessment, this paper endeavours to demonstrate the effectiveness and applicability of the proposed deep learning algorithm in enhancing operational safety, efficiency, and reliability within container terminal management.

1.4. Workflow

Subsequent sections of the workflow analysis will thoroughly examine the existing state-of-the-art by exploring the myriad facets of error detection in container crane operations. Furthermore, methodological approaches employed in addressing this challenge will be scrutinised alongside empirical findings gleaned from relevant studies. This comprehensive analysis will elucidate a nuanced understanding of error prediction within the context of container crane operations, shedding light on its implications and potential avenues for advancement within academic and research discourse.

The following section will inform about the methodology and data used in this research. It presents a novel, custom-built, Python-based algorithm that integrates SMOTE

and LSTM techniques to address imbalanced data through custom preprocessing, analysis, and incorporation of evaluation metrics. The workflow structure is shown in Figure 1 below:

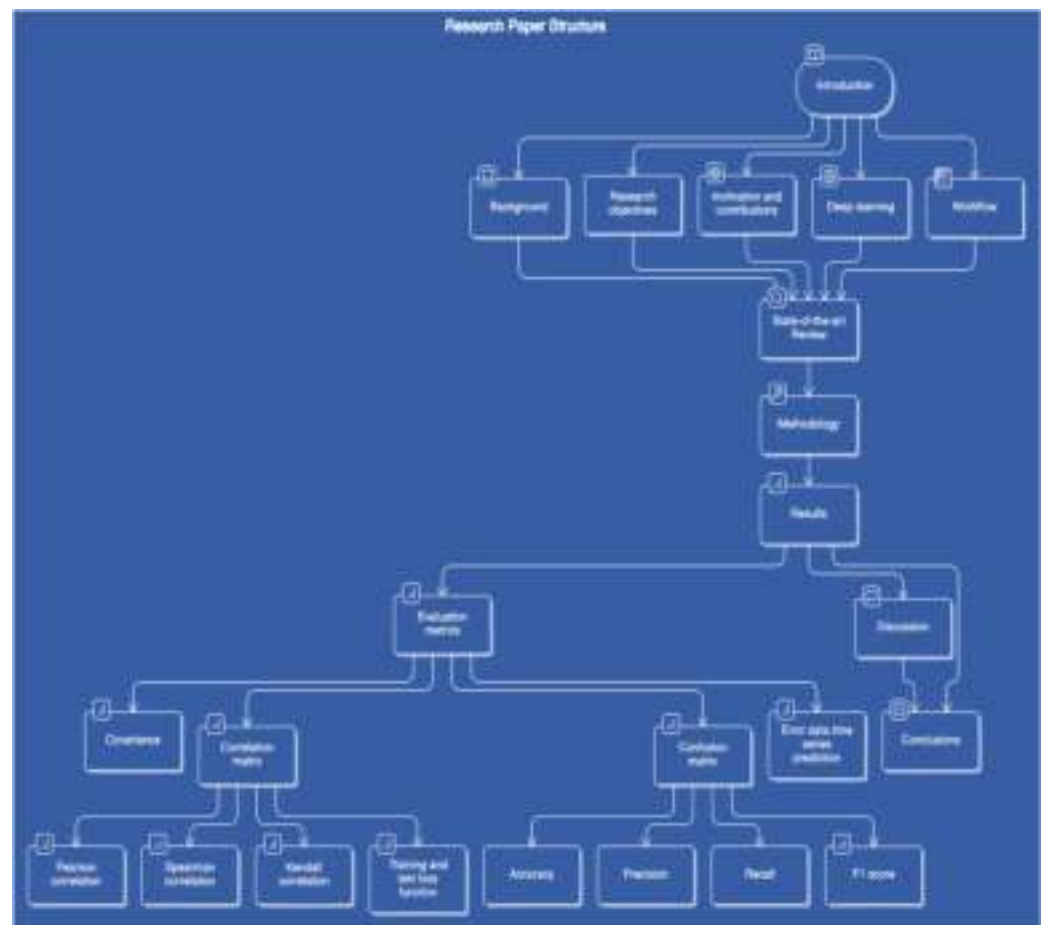


Figure 1. Research paper workflow.

2. State-of-the-Art Review

As highlighted earlier, ensuring the safety and reliability of crane operations is of the utmost importance. Time series data from such heavy machinery operations pose unique challenges due to their large size and complex patterns. Promptly identifying operational errors can reduce accidents and downtime [23]. Machine learning and deep learning models have emerged as powerful tools for detecting anomalies in such data. Extensive literature exists on fault and failure detection in ML [24]. However, very little is available on error detection and prediction. It is crucial to detect and predict an error because an error is a precursor to faults and failure. Figure 2 below illustrates this correlation between error, defect, and failure.

A machine's operational error signal detection refers to the process of identifying abnormal or unexpected signals in a machine's operation. These signals can indicate the system's or machine's problems, malfunctions, or deviations from normal behaviour. This research used actual on-board and off-board STS crane databases from ports and cranes, such as warranty, alarms, and operational data where error signals generated by the crane PLC were available.

The significance of using deep learning for error detection in container crane operations lies in its ability to effectively capture complex patterns, temporal dependencies, and nonlinear relationships inherent in operational data. The vital points highlighting its significance are as follows:

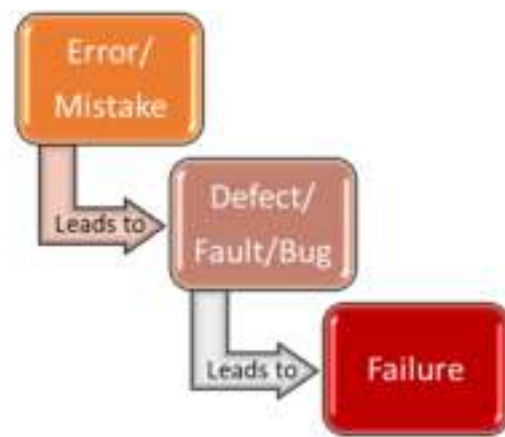


Figure 2. Error as a precursor to failures [25].

1. **Complexity of Operational Data:** Container crane operations generate vast amounts of data, including sensor readings, crane movements, cargo handling, and environmental conditions. Deep learning algorithms, such as RNNs and CNNs, excel at processing high-dimensional and sequential data [18], making them well-suited to analysing the intricate dynamics of crane operations.
2. **Temporal Dependencies:** Crane operations exhibit temporal dependencies, where actions and events at one time step influence subsequent states and outcomes. RNNs, particularly LSTM networks, are designed to capture and learn from temporal dependencies in sequential data, enabling accurate modelling of the temporal evolution of crane operations and the detection of abnormal patterns or deviations [26,27].
3. **Feature Learning:** Deep learning models can automatically learn relevant features from raw data, eliminating the need for manual feature engineering. In the context of error detection in container crane operations, deep learning algorithms can extract informative features from sensor data, crane trajectories, and operational logs, enabling more effective detection of anomalies and deviations indicative of errors.
4. **Adaptability to Complex Environments:** container crane operations are subject to various environmental factors, such as wind, weather conditions, and sea states, which can influence crane performance and introduce uncertainties [28,29]. Deep learning models, with their capacity to learn from diverse and noisy data sources, exhibit robustness and adaptability to complex operating environments, enhancing the reliability and accuracy of error detection systems [22].
5. **Scalability and Generalisation:** Deep learning algorithms are inherently scalable and can handle large volumes of data, making them suitable for analysing the extensive datasets generated by container terminal operations. Moreover, deep learning models can generalise unseen data well, enabling effective error detection across different terminal configurations, crane types, and operating conditions [30].
6. **Continuous Learning and Improvement:** Deep learning models support continuous learning and improvement through iterative training on new data. As additional operational data become available, deep learning-based error detection systems can be updated and refined to adapt to evolving operational conditions, improving their effectiveness over time [16,31].

Several methods have been proposed for error detection in STS container crane operations. The authors of [32] introduced a new method for detecting crane rails' wear, significantly improving accuracy, while [33] identified and assessed human errors among tower crane operators, focusing on the importance of monitoring the anti-collision system. The authors of [34] addressed the issue of detecting physical impacts to the corners of shipping containers during handling operations using the impacts detection methodology. It used the impacts detection methodology to detect recurrent quay crane impacts on the same container locations during hooking. These contemporaneous hits suggest false

hooking, which could cause severe metal deformations over short durations. The study highlights the necessity of shipping container damage detection for port safety and efficiency. In [35], the authors developed a high-speed colour-based object detection algorithm for quayside crane operator assistance systems, significantly increasing the detection rate while maintaining high accuracy. These studies collectively highlight the importance of continuous improvement in error detection methods to enhance safety and efficiency in STS container crane operations. Several studies have demonstrated the efficacy of deep learning for error prediction in container crane operations. Timely error detection is essential in optimising crane performance and guaranteeing safety. Unnoticed errors or problems can compromise production and safety by causing inefficiencies, equipment damage, and possible accidents [36].

Human error is a significant contributing factor in accidents during mobile crane operations. Effective crane control and precise positioning depend on pre-established speeds for each direction, determined according to the crane system's specifications. Detecting problems in speed, acceleration, or deceleration promptly can enhance performance and avoid overloading or crashes. Developing intelligent assistance systems that detect real-time errors can enhance crane operation safety by reducing the likelihood of human errors [37]. Proficient expertise is necessary for the safe operation of overhead cranes and to minimise the occurrence of equipment malfunctions. Nevertheless, the lack of skilled operators has been a worry recently. Early identification of errors, such as problems with rope tension or differences in load weight, can aid in the prevention of accidents and equipment damage, thereby enhancing safety without only depending on the operator's skills [38].

Ensuring prompt identification of errors is essential in successfully implementing efficient automated crane trajectories and providing a user-friendly interface that assists operators in choosing and executing movements. Timely identification of errors in trajectory planning or execution can enhance performance and prevent accidents [36]. The manual planning process currently employed for crane lift path planning is arduous and error-prone, primarily due to construction sites' dynamic and congested conditions. Implementing automated systems capable of real-time error or collision detection can enhance operational efficiency and precision [39].

The research on error prediction in STS container crane operations is diverse, covering various aspects of crane operations. The authors of [40] focused on the fatigue damage analysis of ageing steel structures, a critical factor in error prediction. The authors of [41] identified specific risk factors in stevedoring operations, which can inform error prediction strategies. In [42], the authors addressed the need for robust control systems in offshore container cranes, which can help mitigate errors. The authors of [43] highlighted the importance of structural health monitoring in error prediction, providing a potential method for early detection. Finally, [44] identified a decrease in the mean time between failures due to heavy operations, increased maintenance incidents, and man-hours. Key performance indicators were developed to assess ship-to-shore crane performance and provide insights for future improvements in container crane equipment operation. The findings suggest that the performance indicators can guide future inspections and improvements in container crane operations.

Promising progress is being made in using deep learning for fault identification and prediction in time series data. To tackle problems like noisy labelling, ensemble methods for fault identification [45], and innovative models like GRU-enhanced DCNN for better fault diagnosis in chemical processes [46], several strategies have been put forth. Furthermore, deep learning models for anomaly identification in spacecraft telemetry channels have been investigated, highlighting the necessity of automated methods for effectively monitoring system anomalies [47]. These studies demonstrate how deep learning methods, such as recurrent units and convolutional neural networks, can improve fault detection efficiency, generalisability, and accuracy in various industrial areas.

Deep learning has shown remarkable capabilities in analysing complex data patterns and making predictions. Deep neural networks (DNNs) excel in learning representations

from large-scale data, enabling them to detect subtle signals indicative of impending errors in crane operations. Techniques such as RNNs and CNNs have been adapted to model sequential and spatial dependencies in crane sensor data, operation logs, and environmental factors. Key deep learning architectures that may be used for the Boolean time series data from the STS container cranes include the following [18,48–50]:

- Recurrent Neural Networks (RNNs): RNNs are adept at handling sequential data, rendering them suitable for time-series analysis. RNNs can effectively capture temporal dependencies in crane operation data [51].
- Long Short-Term Memory (LSTM): LSTM networks, a variant of RNNs, are designed to address the vanishing gradient problem. LSTMs are particularly adept at modelling long-term dependencies and have been successfully utilised to predict crane behaviour.
- Gated Recurrent Unit (GRU): GRUs are another RNN variant that maintains memory while being computationally efficient. GRUs have shown promise in predicting input force values for crane control.

RNNs are a robust machine learning technique that excels at handling sequential data, such as time series and natural language processing. RNNs contain a feedback loop that allows them to learn complicated correlations in input sequences and exploit these structures in a nonlinear fashion [52]. RNNs have been extensively employed in diverse domains, such as sentiment analysis, energy demand forecasting, and monthly flow prediction [53]. An essential benefit of RNNs is their capability to handle input sequences of varying lengths and produce output sequences of varying lengths. These characteristics make them suitable for tasks such as language modelling, machine translation, and speech recognition [51]. Deep neural networks, which are capable of learning even more intricate representations of the input data, can be created by stacking RNNs [54].

More sophisticated RNN architectures, such as LSTMs and GRUs, have been created to solve the problem of long-term dependencies, which can be problematic for regular RNNs. These designs better capture long-term dependencies in the input data by employing gating techniques to retain and forget information selectively.

In the available literature, the methodologies used for Boolean time series error or fault detection, diagnosis, and prediction in manufacturing industries based on machine learning can be seen in Table 1 below, and deep learning can be seen in Table 2 below, as follows:

Table 1. Machine learning methodologies.

Methodology	Description	Reference
Fault Detection and Diagnosis (FDD)	These methodologies detect emerging irregularities and predict their consequences for the system's behaviour.	[55]
Data-Driven Fault Diagnosis	This approach uses machine learning algorithms for mechanical fault detection or fault prognosis in manufacturing equipment.	[24]
An array of ML, DL, and DHL Algorithms	These algorithms can potentially perform early fault detection that would lead to future machine failure.	[56]
Automated Machine Learning Approach	This approach is used for real-time fault detection and diagnosis in discrete manufacturing machines.	[57]

Container cranes are crucial in container terminals and are fundamental to the global economy because most of the world's goods pass through them. However, their automation level is lower than those of yard cranes and automated guided vehicles (AGVs) [21]. Various methodologies have been explored here, but the gap lies in further refining and expanding the application of deep learning techniques to address specific challenges in error detection, diagnosis, and prediction in quay container crane data analytics. This would ultimately improve safety, efficiency, and automation in port operations. The essential part is early failure diagnosis, which comes with the initial error identification and prediction step. The proposed methodology aims to do just that.

Table 2. Deep Learning Methodologies.

Methodology	Description	Reference
Deep Learning for Time Series Anomaly Detection	This approach uses specialised deep learning models for detecting anomalous patterns in time series data.	[58]
Deep Generative Models	Variational autoencoder reconstruction along the projection pathway (VAE-RaPP) and a fence generative adversarial network (Fence GAN) are used to deal with imbalanced time series data obtained from manufacturing applications.	[59]
Deep Metric Learning Model	This model considers imbalanced fault data and a quadruplet data pair design manner.	[60]
ATT-1D CNN-GRU Model	This model combines a one-dimensional CNN, GRU, attention mechanism, and KG (knowledge graph) for fault diagnosis.	[61]
Deep-Learning-Based Adaptive Fault Prediction	This approach integrates a comprehensive convolutional feature extraction module, a customised gating module, and a multi-layered progressive extraction module.	[62]

3. Methodology

The dataset used here for deep learning model training and evaluation included the actual data from container cranes at container terminals. It was in a proprietary format and could not be read directly. It was converted into a readable format using Python programming. It consisted of the operational data from the PLC, sensors, and other calculations. The extracted error data were Boolean, meaning the output was 0 or 1.

Software systems are used to diagnose and keep PLC data for 30 days, after which it is discarded. It is a vital tool for crane operations and daily performance evaluation. It provides a dependable data source in time series, raw data directly from the sensors, and their calculations. It detects and locates periodic errors within the crane. It creates approximately 680 to 730 monthly files, storing data per hour. It holds about 23,000 PLC variables in an average crane but can go much higher in a few cranes (e.g., 31,521 PLC variables in STS CC2124). That is 518.4 billion samples per month (20,000 data at 100 ms). The data features were selected (in this case, errors and some other identifier signals), and data from those variables were converted into CSV files in Excel using Python code. The error data contained roughly 3000 variables. Only two columns were non-Boolean, including the last for timestamps/time series (of improper format). There were approximately 3000 columns and more than 16,100,000 (16 million) rows.

The operational dataset analysis assessed eight cranes from different ports; finally, analysing three cranes in detail revealed that most had 2852–2878 error signals. However, only 3 to 6 signals were turning on, showing that an actual error occurred on cranes. This indicated the rarity of error events and the scarcity of relevant data to be analysed further, which posed a significant challenge in the project.

A custom code in Python was developed to extract the data from the proletariat data acquired from the cranes. The resultant Excel files were then used for further data processing.

The custom-built algorithm was developed using SMOTE and LSTM algorithms in Python programming, alongside customised preprocessing procedures, addressing imbalanced data and integrating evaluation metrics.

Data Pre-processing: Customised preprocessing was employed to extract Boolean data. Initially, the unprocessed data, which consisted of time series and sensor readings, were loaded. Ensuring data quality was paramount so that any missing values and outliers were addressed. Subsequently, Boolean features, which consisted of binary flags or true/false values, were extracted.

Addressing Imbalanced Data: In the case of an imbalanced dataset, when one class is considerably underrepresented, SMOTE can be employed. This method produces artificial samples for the underrepresented category by creating new instances through interpolating existing data points. The objective is to achieve a balanced class distribution, guaranteeing equitable representation of both classes. Next, the initial dataset is merged with the

artificially generated samples. Subsequently, the dataset is divided into separate subsets for training and validation purposes. The data were split in the ratio 7:3 for testing the model, with 70% being the training data and the remaining 30% being the testing data. These data were reshaped into a three-dimensional format, with the dimensions being the samples, time steps, and features.

Constructing the LSTM model: Developing the LSTM structure was crucial. An LSTM neural network is built using input layers representing sequential data, LSTM layers capturing temporal dependencies, and an output layer conducting binary classification. The model was compiled by selecting a loss function, such as binary cross-entropy, an optimiser, such as Adam, and an evaluation metric, such as accuracy. The model had a batch size of 50 and was trained for 40 epochs, consisting of 2 LSTM layers and one dense layer for the final output. A seed value was provided at the start, which helped generate reproducible results for the model so that randomly different results were not generated each time the model ran. Once the model was trained, its performance was assessed on the validation set, and any adjustments to the hyperparameters were made, if necessary.

Model Evaluation: The model was evaluated using a range of metrics, such as accuracy, precision, recall, and F1-score. The confusion matrix offers a comprehensive breakdown of predictions, including true positives, true negatives, false positives, and false negatives. The evaluation metrics analysed the model's behaviour and made any required modifications. The custom-built algorithm flowchart is shown below in Figure 3.

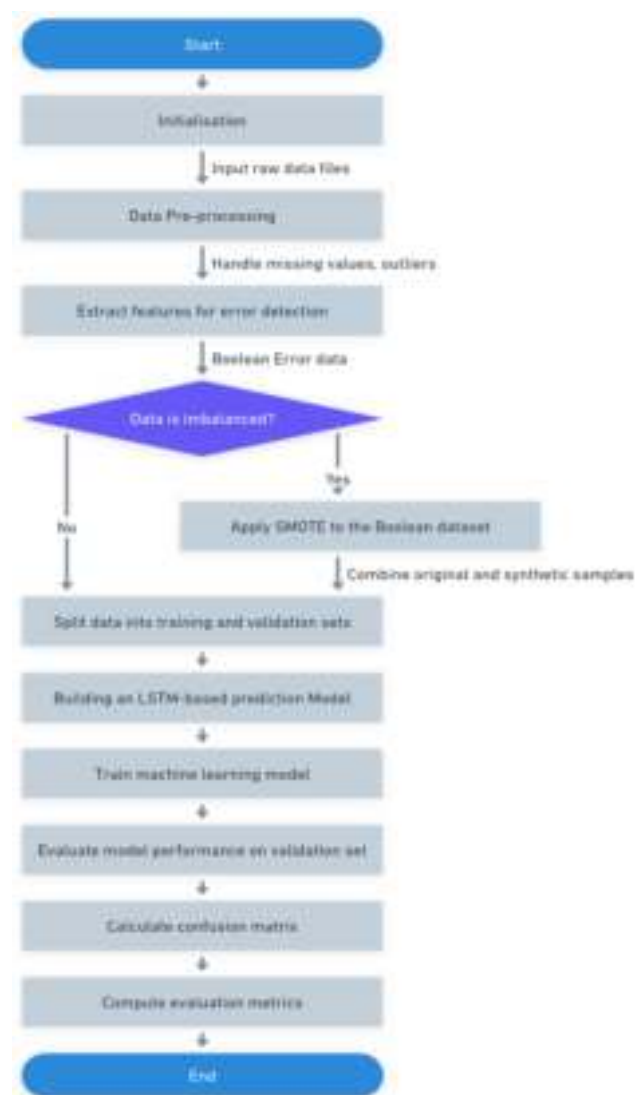


Figure 3. Custom-built algorithm flowchart based on SMOTE and LSTM.

4. Results

This section represents the experimental results of applying the proposed deep learning algorithm to the error detection and prediction task.

4.1. Evaluation Metrics

Evaluation metrics are quantitative measures used to assess the performance of a model or algorithm, particularly in machine learning, statistics, and data analysis. These metrics help determine how well a model is performing and guide improvements and optimisations. The variables shown as var_xxxx in the following metrics represent specific error signals generated from container cranes.

4.1.1. Covariance

Covariance is a consistently additive relationship across the two data samples. This relationship can be summarised between two variables, called the covariance. It is calculated as the average of the product between the values from each sample, where the values have been centred (had their mean subtracted) [63]. Covariance does not imply causation—it only measures the statistical relationship. Understanding covariance can help identify patterns and dependencies when analysing data. A covariance matrix represents the relationships between pairs of variables in a dataset. It quantifies the extent to which two variables vary together.

$$\text{Cov}(X,Y) = (\text{sum}(x - \text{mean}(X)) \times (y - \text{mean}(Y))) \times 1/(n - 1) \quad (1)$$

In this heatmap, each cell corresponds to the covariance between two variables. The diagonal cells (top left to bottom right) represent the covariance of a variable with itself. These values are always positive. The off-diagonal cells (top right and bottom left) represent the covariance between different variables. These values can be positive or negative. The heatmap uses colour intensity to represent covariance values. Dark red indicates a strong positive covariance (close to 1), meaning the variables tend to increase or decrease together. Dark blue indicates a negative covariance (close to −1), suggesting an inverse relationship. Lighter shades represent a weaker correlation (closer to 0).

The heatmap above in Figure 4 shows the x-axis and y-axis labels correspond to specific variables (e.g., 'var_2663' and 'var_831') var_2663 represents .tINTF_General_Visu.tLaneProtection.tCover3.tMotor4.tMotor.tPT100.tAlarm.bStatus and var_831—.tINTF_General_Visu.tLaneProtection.tCover1.tMotor4.tCB.tSignalWithError.tError.bHasError.

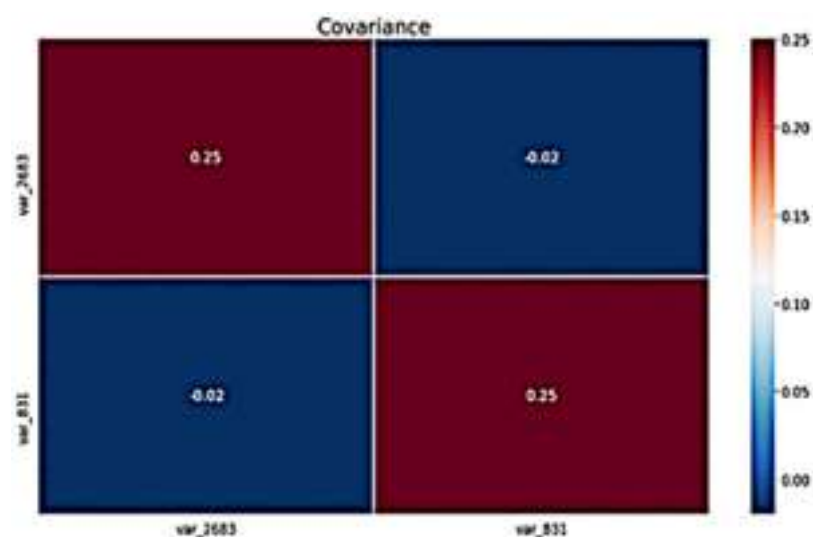


Figure 4. Covariance.

The top left quadrant has a dark red shade, indicating a robust positive covariance (around 0.25) between var_2663 and itself. The bottom right quadrant also shows a solid positive covariance for var_831. The light blue off-diagonal quadrants suggest a weak inverse relationship (around -0.02) between var_2663 and var_831, indicating that motor 4 has a positive covariance with the temperature sensor error in the motor four control block.

4.1.2. Correlation Matrix

The correlation matrix is a frequently used multivariate technique for determining strongly connected variables. This aspect of the project focused on correlating all the variables in the error signals and some operational signals considered dependent variables, such as CraneOn, CraneOff, and twist-lock locked. Error variables with only one unique value, 0, were removed because homogenous variables contain only one unique value and do not contribute additional information to the model. The correlation between the variables was analysed and visualised to give more insight into the data. Correlation was employed to determine the relationship between variables. It can be positive, indicating that both variables move in the same direction, or negative, indicating that when one variable's value increases, the values of the other variables decrease. Correlation can also be neutral or zero, indicating that the variables have no relationship or are unrelated.

Pearson Correlation

Pearson's correlation coefficient is computed by dividing the covariance of two variables by the product of their standard deviations. It is the normalisation of the covariance between the two continuous variables to give an interpretable score. It quantifies the degree to which a change in one variable is associated with a change in another variable [63].

$$\text{PearsonCorrelation}(r) = \text{covariance}(X,Y) / (\text{stdv}(X) \times \text{stdv}(Y)) \quad (2)$$

This heatmap shown in Figure 5 above, compares the two variables var_2683, i.e., .tINTF_General_Visu.tLaneProtection.tCover3.tMotor4.tMotor.tPT100_Group.tPT100_2.tError.bHasError and "var_831" i.e., .tINTF_General_Visu.tLaneProtection.tCover1.tMotor4.tCB.tSignalWithError.tError.bHasError. The heatmap displays the correlation coefficient between these variables. The diagonal quadrants (where a variable is compared to itself) show a correlation coefficient 1 (as expected). The off-diagonal quadrants reveal a weak negative correlation of approximately -0.079 . The colour scale ranges from -1 to 1 . Dark red represents stronger negative correlations, while dark blue indicates stronger positive correlations.

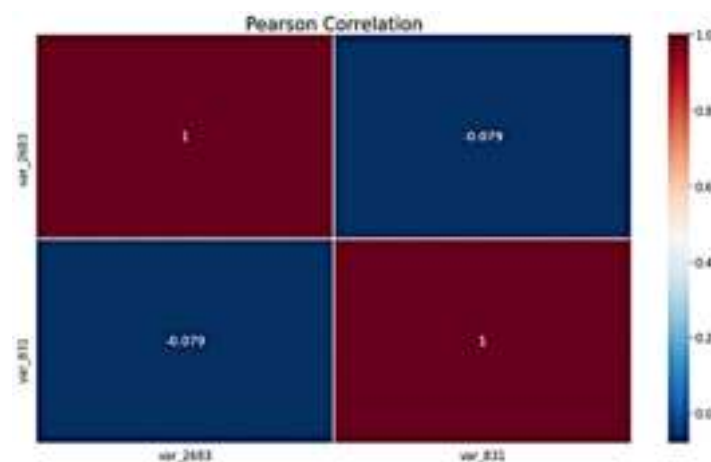


Figure 5. Pearson's correlation.

Spearman Correlation

Spearman's correlation was calculated from each sample's relative rank of values. This approach is commonly used in non-parametric statistics, such as statistical methods where no specific distribution of the data, like the Gaussian distribution, is assumed [64]. This coefficient measures the strength and direction of the monotonic relationship between two ranked variables. It is beneficial in identifying and assessing nonlinear relationships and ordinal data.

$$\text{SpearmanCorrelation} = \text{covariance}(\text{rank}(X), \text{rank}(Y)) / \text{std}(\text{rank}(X)) \times \text{std}(\text{rank}(Y)) \quad (3)$$

This heatmap shown above in Figure 6, is divided into four quadrants to show the Spearman correlation coefficient between two variables labelled as var_2683—tINTF_General_Visu.tLaneProtection.tCover3.tMotor4.tMotor.tPT100_Group.tPT100_2.tError.bHasError on the y-axis and var_831—tINTF_General_Visu.tLaneProtection.tCover1.tMotor4.tCB.tSignalWithError.tError.bHasError on the x-axis. The two diagonal quadrants show a perfect positive correlation (a value of 1) for each variable with itself (indicated by the red colour and the number 1). The off-diagonal quadrants reveal a very low negative correlation of -0.079 between the two variables (indicated by deep blue).

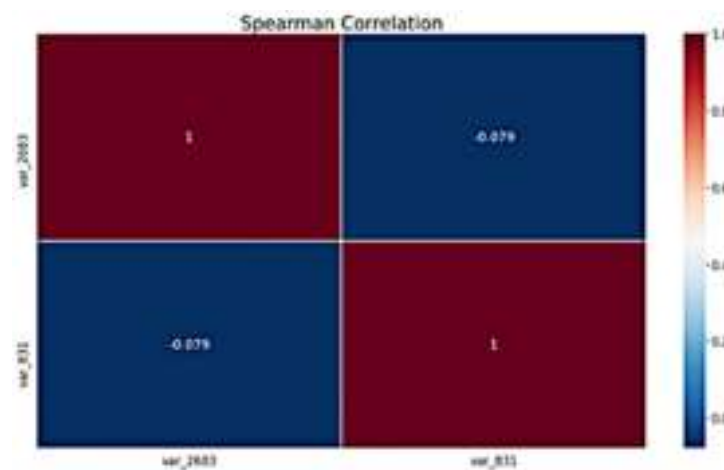


Figure 6. Spearman correlation.

Kendall Correlation

The Kendall Tau rank correlation coefficient measures the degree of similarity between two sets of ranks given to the same set of objects. Unlike Spearman's coefficient, the Kendall Tau measures only directional agreement, not rank differences.

$$\text{KendallCorrelation} = (\text{Number of Concordant Pairs} - \text{Number of Discordant pairs}) / n(n - 1) / 2 \quad (4)$$

The Kendall correlation heatmap displayed in Figure 7 reaffirms the findings of the other correlations. It focuses on the same two variables as the other correlation heatmaps, i.e., var_2683—tINTF_General_Visu.tLaneProtection.tCover3.tMotor4.tMotor.tPT100_Group.tPT100_2.tError.bHasError on the y-axis and var_831—tINTF_General_Visu.tLaneProtection.tCover1.tMotor4.tCB.tSignalWithError.tError.bHasError on the x-axis. The top left and bottom right quadrants are dark red, indicating a perfect correlation of 1. This is expected since each variable correlates with itself. The top right and bottom left quadrants are blue, suggesting a weak negative correlation (approximately -0.079) between var_2683 and var_831.

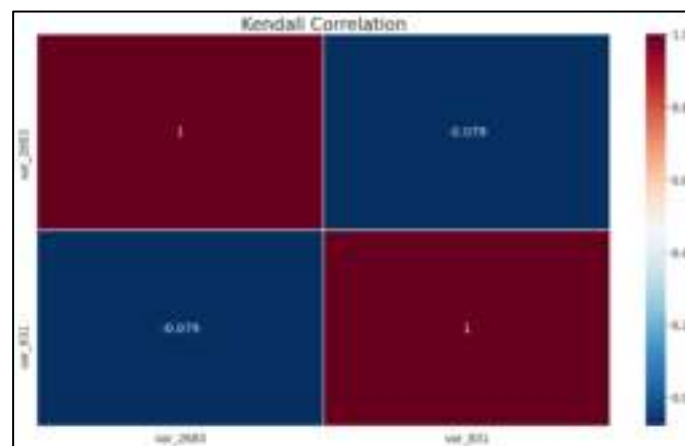


Figure 7. Kendall correlation.

4.1.3. Training and Test Loss Function

In machine learning, the training loss and test loss (or validation loss) are critical metrics used to evaluate a model's performance during the training process. These loss functions help measure how well the model performs on the training and unseen data (test or validation data). Training loss is the error or discrepancy between the predicted and actual outputs on the training dataset. It provides a measure of how well the model fits the training data. Test loss (or validation loss) is the error between the predicted and actual test or validation dataset outputs. This dataset is not used during training but is kept aside to evaluate the model's performance on unseen data.

The root mean squared error (RMSE) value is calculated. The square root of the mean squared error provides a measure in the same units as the target variable.

The graph in Figure 8 shows two lines: train (in blue) and test (in orange). The horizontal axis represents the number of epochs, ranging from 0 to 40, while the vertical axis represents the loss, ranging from 0.0 to 0.7. Both lines initially show a sharp decline in loss during the early epochs, indicating effective learning from the training data. As the epochs progress, both lines gradually flatten out, suggesting diminished returns on learning and the risk of overfitting if the test loss increases. The train line represents the loss of the training data, and the test line represents the loss of unseen test data. Monitoring both lines helps assess model performance and generalisation ability and aids in diagnosing issues like overfitting, underfitting, and convergence. This graph is academically relevant in evaluating ML models and is used to optimise training strategies and improve model performance.

4.1.4. Confusion Matrix

A confusion matrix is a fundamental tool for evaluating the performance of classification models. It provides a comprehensive view of how well a model predicts different classes. The confusion matrix is based on a binary classification problem with two classes: 0 and 1. These classes could represent various outcomes, such as pass/fail, yes/no, or true/false. The matrix is organised into four quadrants:

- True Negatives (TN): Instances correctly predicted as class 0. These are desirable predictions—correctly identifying instances as negative (class 0);
- False Positives (FP): Instances incorrectly predicted as class 1 when they are actually class 0. These are errors—instances wrongly classified as positive (class 1) when they are not;
- False Negatives (FN): Instances incorrectly predicted as class 0 when they are actually class 1. These are also errors—instances wrongly classified as negative when they are positive;

- True Positives (TP): Instances correctly predicted as class 1. These are desirable predictions—correctly identifying instances as positive.



Figure 8. The training and test loss function.

The confusion matrix shown in Figure 9 summarises the classification model's performance. It provides insights into how well the model's predictions aligned with the actual outcomes.

1. True Negative (TN): The model correctly predicted negative instances (the actual outcome was negative). There are 997 true negatives;
2. False Positive (FP): The model incorrectly predicted positive instances (the actual negative outcome). Here, there are nine false positives;
3. False Negative (FN): The model incorrectly predicted negative instances (the actual positive outcome). The matrix shows four false negatives;
4. True Positive (TP): The model correctly predicted positive instances (the actual outcome was positive). In this matrix, there are four true positives.

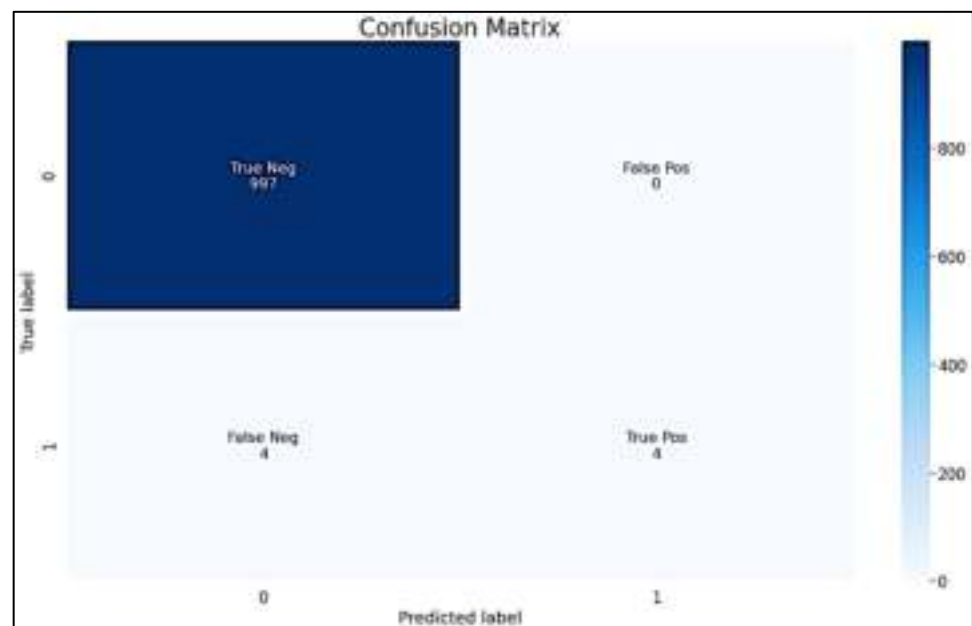


Figure 9. The confusion matrix.

4.1.5. Accuracy, Precision, Recall, F1 Score

The accuracy, precision, recall, and F1 score shown in Figure 10 below were also calculated and printed for the predicted data evaluation metrics. These model performances were then saved in a text file.

```
Accuracy: 0.996
Precision: 1.000
Recall: 0.500
F1 score: 0.667
```

Figure 10. Evaluation metrics results.

Accuracy: The number of correct predictions is divided by the total number of predictions.

$$\text{Accuracy} = (\text{True Positives} + \text{True Negatives}) / \text{Total Instances} \quad (5)$$

Precision: This measures the accuracy of positive class predictions by determining how many of them actually belong to the positive class.

$$\text{Precision} = \text{True Positives} / (\text{True Positives} + \text{False Positives}) \quad (6)$$

Recall (Sensitivity): This calculates the proportion of correct positive predictions out of all the positive examples in the dataset.

$$\text{Recall} = \text{True Positives} / (\text{True Positives} + \text{False Negatives}) \quad (7)$$

F1 Score: This provides a single score that balances precision and recall in one number. It is a harmonic mean of precision and recall.

$$\text{F1 Score} = 2 \times ((\text{Precision} \times \text{Recall}) / (\text{Precision} + \text{Recall})) \quad (8)$$

Accuracy: Achieving an accuracy of 0.996 indicates overall solid performance.

Precision: A precision score 1.00 suggests that the model rarely misclassified true errors as non-errors.

Recall: The recall score 0.500 indicates that the model identified only half of the actual errors. This trade-off between precision and recall is common in imbalanced datasets.

F1 Score: The F1 score (0.667) balances precision and recall, providing a comprehensive view of the model's effectiveness.

4.2. Error Data Time Series Prediction

The final result is the prediction of the following n seconds of data where the user provides the n seconds as input in the beginning, and the corresponding data are saved in a CSV file. For this, the next 1 s worth of data is predicted for the next 60 s with the help of historical data available for analysis. The output from the predicted CSV file for one of the error variables is shown below in Table 3:

Table 3. Predicted error data.

Timestamp	var_831
12/11/2020 20:35	0
12/11/2020 20:35	0
12/11/2020 20:35	0
12/11/2020 20:35	0
12/11/2020 20:35	0

Table 3. Cont.

Timestamp	var_831
12/11/2020 20:35	0
12/11/2020 20:35	0
12/11/2020 20:35	0
12/11/2020 20:35	0
12/11/2020 20:35	0
12/11/2020 20:35	0
12/11/2020 20:35	0
12/11/2020 20:35	0
12/11/2020 20:35	0
12/11/2020 20:35	0
12/11/2020 20:35	0
12/11/2020 20:35	0
12/11/2020 20:35	0
12/11/2020 20:35	0
12/11/2020 20:35	0
12/11/2020 20:35	0
12/11/2020 20:35	0
12/11/2020 20:35	1
12/11/2020 20:35	1
12/11/2020 20:35	1
12/11/2020 20:35	0
12/11/2020 20:35	0
12/11/2020 20:35	0
12/11/2020 20:35	0
12/11/2020 20:35	1
12/11/2020 20:35	1
12/11/2020 20:35	1
12/11/2020 20:36	1
12/11/2020 20:36	1
12/11/2020 20:36	0
12/11/2020 20:36	0
12/11/2020 20:36	0
12/11/2020 20:36	0
12/11/2020 20:36	1
12/11/2020 20:36	1
12/11/2020 20:36	1
12/11/2020 20:36	1
12/11/2020 20:36	1
12/11/2020 20:36	0
12/11/2020 20:36	0
12/11/2020 20:36	0
12/11/2020 20:36	0

Table 3. Cont.

Timestamp	var_831
12/11/2020 20:36	0
12/11/2020 20:36	0
12/11/2020 20:36	0
12/11/2020 20:36	0
12/11/2020 20:36	0
12/11/2020 20:36	0
12/11/2020 20:36	0
12/11/2020 20:36	0
12/11/2020 20:36	0
12/11/2020 20:36	0
12/11/2020 20:36	0
12/11/2020 20:36	0
12/11/2020 20:36	0
12/11/2020 20:36	0
12/11/2020 20:36	0
12/11/2020 20:36	0
12/11/2020 20:36	0
12/11/2020 20:36	0

5. Discussion

Timely error detection and prediction are essential in optimising crane performance. They are crucial in ensuring safety by preventing accidents, protecting equipment from damage, and reducing human errors. As a result, these practices enhance productivity and lower operating expenses.

Understanding predictive analytics' limitations is critical. First and foremost, progress is impossible without an extensive and high-quality training dataset. Second, having a clear description of the concept to be forecasted and past occurrences is critical. One of the challenges in predictive analytics is identifying a needle in a haystack and predicting its occurrence. However, there are certain limitations of traditional methods. These limitations in container crane operations are as follows:

1. **Human-Induced Errors:** Human operators still play a significant role in quay crane operations despite automation advancements. Their decisions and actions can lead to errors, affecting productivity and safety [64].
2. **Complex Dynamics:** Container cranes operate in a dynamic environment with varying container weights, wind conditions, and vessel movements. Predicting errors in such a complex system is challenging [65–67].
3. **Efficiency and Downtime:** Minimising downtime due to errors is critical. Timely error prediction can prevent costly delays and optimise crane utilisation [68,69].
4. **Imbalanced data:** A dataset is called imbalanced when there are substantial disparities in the number of examples of each class, resulting in one or more classes being underrepresented in the dataset [70]. This phenomenon has been established in several real-world datasets gathered in industrial settings.

Therefore, this research developed a novel, custom-built algorithm to address these limitations and challenges by focusing on detecting and predicting errors before they convert into faults and failures.

The following points address any challenges and limitations encountered during the implementation:

1. Because there was a significant time difference between the data array files, the data array files could not be merged for analysis; instead, analysis was performed on each data array file separately.

2. For time series forecasting, the data were scaled down from milliseconds to seconds, because the data in milliseconds were not continuous, i.e., (1, 2, 3, ..., 999).
3. There may be a good chance of error as the dataset size was small, so the LSTM model could not properly and correctly predict new data. By adding more data, this issue can be resolved.

Despite the promising results, implementing deep learning models for error prediction in real-world crane operations remains challenging. These include data quality issues, the interpretability of complex models, and integration with existing systems. Future research directions may address these challenges through advanced model architectures, data fusion techniques, and collaborative efforts between academia and industry stakeholders. The avenues for future research can advance with improved data quality and quantity, which are currently lagging.

6. Conclusions

Research on container cranes encompasses various aspects related to their design, operation, and impact on port logistics. However, there is a glaring absence of studies focusing on early error identification to prevent faults and failures. Timely error prediction through deep learning algorithms has the potential to revolutionise crane performance and ensure optimal efficiency within container crane operations.

A custom-built deep learning algorithm based on LSTM was developed, achieving an accuracy of 99.6% in predicting crane error signals using LSTM and SMOTE for classification prediction, with a precision of 1.00, recall of 0.500, and an f1 score of 0.667. However, further data are required to improve the model's outcomes.

The current data availability is limited to a month with approximately 30,000 variables, including less than 3000 error variables. The usable data were a small sample set, with only 3 to 6 error variables that were occasionally activated. The study also provides other outputs, such as correlation, covariance, and confusion matrices. The experiments demonstrated the technique's capability to identify crucial elements, leading to efforts to enhance the model's structure for improved error prediction in industrial big data.

While the output model relies on pre-existing data, its ability to handle imbalanced data, a common issue in most industries, presents a significant advantage. With further data, the existing model can be enhanced. Continuous data over several months will provide a more comprehensive error data sample set, contributing to improved outputs. Once a larger dataset is obtained in future, exploration of other algorithms such as random forest, naïve Bayes, and XG boost may be used.

Continued research and innovation in this area are essential to fully realising the benefits of predictive analytics in port operations.

Author Contributions: Conceptualization, A.A.; data curation, A.A.; formal analysis, A.A.; methodology, A.A.; software, A.A.; supervision, L.K. and J.W.; validation, A.A.; visualization, A.A.; writing—original draft, A.A.; writing—review and editing, A.A., L.K. and J.W. All authors have read and agreed to the published version of the manuscript.

Funding: This project was supported, in part, by Science Foundation Ireland grant 13/RC/2094_P2 and co-funded under the European Regional Development Fund through the Southern & Eastern Regional Operational Programme to Lero—the Science Foundation Ireland Research Centre for Software (www.lero.ie, accessed on 20 July 2024) during the initial two years of this project.

Data Availability Statement: The datasets presented in this article are not readily available because of the container port's privacy or data security issues. Requests to access the datasets should be directed to the corresponding author of this paper.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Mordor Intelligence. Container Shipping Market—Size, Share & Growth Analysis. *Mordor Intelligence Industry Reports*. [Online]. Available online: <https://www.mordorintelligence.com/industry-reports/global-container-shipping-market> (accessed on 17 July 2024).
2. Lindeberg, E.; Improving the Operational Performance of STS Cranes. Västerås, Sweden, February 2011. Available online: www.porttechnology.org (accessed on 18 June 2021).
3. Ship-to-Shore (STS) Container Cranes Market Size, Share, Industry Forecast 2028. Available online: <https://www.fortunebusinessinsights.com/ship-to-shore-sts-container-cranes-market-102880> (accessed on 15 June 2021).
4. Zrniü, N.; Hoffmann, K. Development of Design of Ship-To-Shore Container Cranes: 1959–2004. In *International Symposium on History of Machines and Mechanisms*; Springer: Dordrecht, The Netherlands, 2007. [CrossRef]
5. Khoukhi, A.; Khalid, M.H. Hybrid computing techniques for fault detection and isolation, a review. *Comput. Electr. Eng.* **2015**, *43*, 17–32. [CrossRef]
6. Saltzer, J.H.; Frans Kaashoek, M. 2.2: Faults, Failures, and Fault-Tolerant Design 2.2.1: Faults, Failures, and Modules. In *Principles of Computer System Design*; Massachusetts Institute of Technology LibreTexts: Davis, CA, USA, 2024; Chapter 2.2, pp. 1–461. Available online: <https://eng.libretexts.org/@go/page/58498> (accessed on 19 July 2024).
7. Safaei, M.M.; Hejazian, S.; Pedrammehr, S.; Pakzad, M.; Etefagh, M.; Fotouhi, M. Damage Detection of Gantry Crane with a Moving Mass Using Artificial Neural Network. *Buildings* **2024**, *14*, 458. [CrossRef]
8. Wang, Z.; Cheng, J.; Hu, H. A Proactive-Reactive-Based Approach for Continuous Berth Allocation and Quay Crane Assignment Problems with Hybrid Uncertainty. *J. Mar. Sci. Eng.* **2024**, *12*, 182. [CrossRef]
9. Gattuso, D.; Pratico, F.G.; Longo, R.; Cassone, G.; Vigna, M.; Sceni, R. Rail operations in freight terminals: Safety issues and proposed methodology. In *5th IEEE International Conference on Models and Technologies for Intelligent Transportation Systems, MT-ITS 2017—Proceedings, Naples, Italy, 26–28 June 2017*; Institute of Electrical and Electronics Engineers Inc.: Piscataway, NJ, USA, 2017; pp. 164–169. [CrossRef]
10. Huang, M.; Liu, Z.; Tao, Y. Mechanical fault diagnosis and prediction in IoT based on multi-source sensing data fusion. *Simul. Model. Pract. Theory* **2020**, *102*, 101981. [CrossRef]
11. Meddaoui, A.; Hain, M.; Hachmoud, A. The benefits of predictive maintenance in manufacturing excellence: A case study to establish reliable methods for predicting failures. *Int. J. Adv. Manuf. Technol.* **2023**, *128*, 3685–3690. [CrossRef]
12. Fernandes, M.; Canito, A.; Bolón-Canedo, V.; Conceição, L.; Praça, I.; Marreiros, G. Data analysis and feature selection for predictive maintenance: A case-study in the metallurgic industry. *Int. J. Inf. Manag.* **2019**, *46*, 252–262. [CrossRef]
13. Yeoh, J.K.W.; Wong, J.H.; Peng, L. Integrating Crane Information Models in BIM for Checking the Compliance of Lifting Plan Requirements. In *Proceedings of the 33rd International Symposium on Automation and Robotics in Construction (ISARC 2016)*, Auburn, AL, USA, 18–21 July 2016; Available online: <https://www.iaarc.org/publications/fulltext/ISARC2016-Paper192.pdf> (accessed on 20 July 2024).
14. Chu, F.; Gailus, S.; Liu, L.; Ni, L. The Future of Port Automation | McKinsey. 2018. Available online: <https://www.mckinsey.com/industries/travel-logistics-and-infrastructure/our-insights/the-future-of-automated-ports> (accessed on 18 May 2024).
15. Lepenioti, K.; Pertselakis, M.; Bousdekis, A.; Louca, A.; Lampathaki, F.; Apostolou, D.; Mentzas, G.; Anastasiou, S. Machine Learning for Predictive and Prescriptive Analytics of Operational Data in Smart Manufacturing. In *Advanced Information Systems Engineering Workshops*; Springer: Cham, Switzerland, 2020. [CrossRef]
16. What Is Deep Learning? | A Beginner's Guide. Available online: <https://www.scribbr.com/ai-tools/deep-learning/> (accessed on 19 May 2024).
17. Zhao, R.; Yan, R.; Chen, Z.; Mao, K.; Wang, P.; Gao, R.X. Deep learning and its applications to machine health monitoring. *Mech. Syst. Signal Process.* **2019**, *115*, 213–237. [CrossRef]
18. Kumar, A.; Gaur, N.; Chakravarty, S.; Alsharif, M.H.; Uthansakul, P.; Uthansakul, M. Analysis of spectrum sensing using deep learning algorithms: CNNs and RNNs. *Ain Shams Eng. J.* **2024**, *15*, 102505. [CrossRef]
19. Ubal, C.; Di-Giorgi, G.; Contreras-Reyes, J.E.; Salas, R. Predicting the Long-Term Dependencies in Time Series Using Recurrent Artificial Neural Networks. *Mach. Learn. Knowl. Extr.* **2023**, *5*, 1340–1358. [CrossRef]
20. Heilig, L.; Stahlbock, R.; Voß, S. From Digitalization to Data-Driven Decision Making in Container Terminals. *arXiv* **2019**, arXiv:1904.13251.
21. Shin, H.-S.; Lee, S.-P.; Ha, Y.-S.; Kim, H.-S. Designing container crane control learning model using deep learning. *J. Adv. Mar. Eng. Technol.* **2023**, *47*, 2023–2234. [CrossRef]
22. Chalapathy, R.; Chawla, S. Deep Learning for Anomaly Detection: A Survey. *arXiv* **2019**, arXiv:1901.03407.
23. Shagluf, A.; Longstaff, A.P.; Fletcher, S. (PDF) Maintenance Strategies to Reduce Downtime Due to Machine Positional Errors, Maintenance Performance Measurement and Management (MPMM). Available online: https://www.researchgate.net/publication/267864966_Maintenance_Strategies_to_Reduce_Downtime_Due_to_Machine_Positional_Errors (accessed on 18 July 2024).
24. Fernandes, M.; Corchado, J.M.; Marreiros, G. Machine learning techniques applied to mechanical fault diagnosis and fault prognosis in the context of real industrial manufacturing use-cases: A systematic literature review. *Appl. Intell.* **2022**, *52*, 14246–14280. [CrossRef] [PubMed]

25. Software Testing—Bug vs Defect vs Error vs Fault vs Failure—GeeksforGeeks. Available online: <https://www.geeksforgeeks.org/software-testing-bug-vs-defect-vs-error-vs-fault-vs-failure/> (accessed on 22 April 2024).
26. Jalayer, M.; Orsenigo, C.; Vercellis, C. Fault detection and diagnosis for rotating machinery: A model based on convolutional LSTM, Fast Fourier and continuous wavelet transforms. *Comput. Ind.* **2021**, *125*, 103378. [CrossRef]
27. Zhao, R.; Yan, R.; Wang, J.; Mao, K. Learning to monitor machine health with convolutional Bi-directional LSTM networks. *Sensors* **2017**, *17*, 273. [CrossRef] [PubMed]
28. Zrnic, N.; Petković, Z.; Bošnjak, S.M. Automation of ship-to-shore container cranes: A review of state-of-the-art. *FME Transactions* **2005**, *33*, 111–121. Available online: https://www.researchgate.net/publication/228936324_Automation_of_ship-to-shore_container_cranes_A_review_of_state-of-the-art (accessed on 24 March 2024).
29. Tran, Q.H.; Huh, J.; Nguyen, V.B.; Kang, C.; Ahn, J.H.; Park, I.J. Sensitivity analysis for ship-to-shore container crane design. *Appl. Sci.* **2018**, *8*, 1667. [CrossRef]
30. Li, Z. Deep Learning Driven Approaches for Predictive Maintenance A Framework of Intelligent Fault Diagnosis and Prognosis in the Industry 4.0 Era. Ph.D. Thesis, Norwegian University of Science and Technology, Trondheim, Norway, 2018.
31. Goodfellow, I.; Bengio, Y.; Courville, A. 2015. Available online: <http://www.deeplearningbook.org/> (accessed on 15 June 2024).
32. Liu, W.; Xiao, J. A new method of detection the wear of crane rails and its error analysis. *J. Mech. Strength* **2014**, *36*, 878–883. Available online: https://www.researchgate.net/publication/293095272_A_new_method_of_detection_the_wear_of_crane_rails_and_its_error_analysis (accessed on 19 March 2024).
33. Mohammadfam, I.; Borgheipour, H.; Tehrani, G.; Madadi, S. Identification and assessment of human errors among tower crane operators using SHERPA and CREAM techniques. *J. Health Saf. Work.* **2020**, *10*, 5–8.
34. Jakovlev, S.; Eglynas, T.; Jusis, M.; Voznak, M.; Partila, P.; Tovarek, J. Detecting Physical Impacts to the Corners of Shipping Containers during Handling Operations Performed by Quay Cranes. *J. Mar. Sci. Eng.* **2023**, *11*, 794. [CrossRef]
35. Gao, X.; Yeh, H.G.; Marayong, P. A high-speed color-based object detection algorithm for quayside crane operator assistance system. In Proceedings of the 11th Annual IEEE International Systems Conference, SysCon 2017—Proceedings, Montreal, QC, Canada, 24–27 April 2017. [CrossRef]
36. Westerberg, S.; Shiriaev, A. Virtual Environment-Based Teleoperation of Forestry Machines: Designing Future Interaction Methods. *J. Hum.-Robot. Interact.* **2013**, *2*, 84–110. [CrossRef]
37. Ren, W.; Wu, Z.; Zhang, L. Real-time planning of a lifting scheme in mobile crane mounted controllers. *Can. J. Civ. Eng.* **2016**, *43*, 542–552. [CrossRef]
38. Watanabe, M.; Momoi, Y.; Odai, M.; Ieshige, K. Simultaneous estimation for rope tension and load of cranes using motor driving signals. *Trans. JSME* **2023**, *89*. [CrossRef]
39. Lei, Z.; Behzadipour, S.; Al-Hussein, M.; Hermann, U. Application of robotic obstacle avoidance in crane lift path planning. In Proceedings of the 28th ISARC, Seoul, Republic of Korea, 29 June–2 July 2011.
40. Lehner, P.; Krivý, V.; Krejsa, M.; Pařenica, P.; Kozák, J. Stochastic Service Life Prediction of Existing Steel Structure Loaded by Overhead Cranes. *Procedia Struct. Integr.* **2018**, *13*, 1539–1544. [CrossRef]
41. Shang, K.C.; Tseng, W.J. A risk analysis of stevedoring operations in seaport container terminals. *J. Mar. Sci. Technol.* **2010**, *18*, 201–210. [CrossRef]
42. Ismail, R.M.T.R.; That, N.D.; Ha, Q.P. Modelling and robust trajectory following for offshore container crane systems. *Autom. Constr.* **2015**, *59*, 179–187. [CrossRef]
43. Liu, J.; Qin, X.; Sun, Y.; Zhang, Q. Response Prediction Model for Structures of Quayside Container Crane Based on Monitoring Data. *J. Perform. Constr. Facil.* **2021**, *35*. [CrossRef]
44. Jo, J.H.; Kim, S. Key Performance Indicator Development for Ship-to-Shore Crane Performance Assessment in Container Terminal Operations. *J. Mar. Sci. Eng.* **2020**, *8*, 6. [CrossRef]
45. Cheng, C.; Liu, X.; Zhou, B.; Yuan, Y. Intelligent Fault Diagnosis With Noisy Labels via Semisupervised Learning on Industrial Time Series. *IEEE Trans. Ind. Inform.* **2023**, *19*, 7724–7732. [CrossRef]
46. Lee, X.Y.; Kumar, A.; Vidyaratne, L.; Rao, A.R.; Farahat, A.; Gupta, C. An ensemble of convolution-based methods for fault detection using vibration signals. In Proceedings of the 2023 IEEE International Conference on Prognostics and Health Management (ICPHM 2023), Montreal, QC, Canada, 5–7 June 2023; pp. 172–179. [CrossRef]
47. Zhang, J.; Zhang, M.; Feng, Z.; Ruifang, L.V.; Lu, C.; Dai, Y.; Dong, L. Gated recurrent unit-enhanced deep convolutional neural network for real-time industrial process fault diagnosis. *Process Saf. Environ. Prot.* **2023**, *175*, 129–149. [CrossRef]
48. How To Code RNN and LSTM Neural Networks in Python. Available online: <https://www.nbshare.io/notebook/249468051/How-To-Code-RNN-and-LSTM-Neural-Networks-in-Python/> (accessed on 28 April 2022).
49. Quiñones-Grueiro, M.; Llanes-Santiago, O.; Silva Neto, A.J. Fault Diagnosis in Industrial Systems. *Stud. Syst. Decis. Control* **2021**, *309*, 1–14. [CrossRef]
50. Illustrated Guide to LSTM’s and GRU’s: A Step by Step Explanation | by Michael Phi | towards Data Science. Available online: <https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21> (accessed on 13 April 2021).
51. Maheswaranathan, N.; Williams, A.H.; Golub, M.D.; Ganguli, S.; Sussillo, D. Reverse engineering recurrent networks for sentiment classification reveals line attractor dynamics. In Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, BC, Canada, 8–14 December 2019; pp. 15696–15705.

52. Alizadeh, Z.; Yazdi, J.; Kim, J.H.; Al-Shamiri, A.K. Assessment of Machine Learning Techniques for Monthly Flow Prediction. *Water* **2018**, *10*, 1676. [CrossRef]
53. Ahmad, J.; Tahir, A.; Larijani, H.; Ahmed, F.; Aziz Shah, S.; Hall, A.J.; Buchanan, W.J. Energy demand forecasting of buildings using random neural networks. *J. Intell. Fuzzy Syst.* **2020**, *38*, 4753–4765. [CrossRef]
54. Li, S.; Li, W.; Member, S.; Cook, C.; Gao, Y. Deep Independently Recurrent Neural Network (IndRNN). *arXiv* **2019**, arXiv:1910.06251v3.
55. Abid, A.; Khan, M.T.; Iqbal, J. A review on fault detection and diagnosis techniques: Basics and beyond. *Artif. Intell. Rev.* **2021**, *54*, 3639–3664. [CrossRef]
56. Shahin, M.; Chen, F.F.; Hosseinzadeh, A.; Zand, N. Using machine learning and deep learning algorithms for downtime minimization in manufacturing systems: An early failure detection diagnostic service. *Int. J. Adv. Manuf. Technol.* **2023**, *128*, 3857–3883. [CrossRef]
57. Leite, D.; Martins, A.; Rativa, D.; De Oliveira, J.F.L.; Maciel, A.M.A. An Automated Machine Learning Approach for Real-Time Fault Detection and Diagnosis. *Sensors* **2022**, *22*, 6138. [CrossRef]
58. Darban, Z.Z.; Webb, G.L.; Pan, S.; Aggarwal, C.C.; Salehi, M. Deep Learning for Time Series Anomaly Detection: A Survey. *arXiv* **2022**, arXiv:2211.05244v2.
59. Song, J.; Lee, Y.C.; Lee, J. Deep generative model with time series-image encoding for manufacturing fault detection in die casting process. *J. Intell. Manuf.* **2023**, *34*, 3001–3014. [CrossRef]
60. Gui, X.; Zhang, J. Deep Metric Learning Model for Imbalanced Fault Diagnosis. *arXiv* **2021**, arXiv:2107.03786.
61. Liu, H.; Ma, R.; Li, D.; Yan, L.; Ma, Z. Machinery Fault Diagnosis Based on Deep Learning for Time Series Analysis and Knowledge Graphs. *J. Signal Process. Syst.* **2021**, *93*, 1433–1455. [CrossRef]
62. Pang, J.L. Adaptive fault prediction and maintenance in production lines using deep learning. *Int. J. Simul. Model.* **2023**, *22*, 734–745. [CrossRef]
63. Brownlee, J.; How to Calculate Correlation Between Variables in Python. Machine Learning Mastery. Available online: <https://machinelearningmastery.com/how-to-use-correlation-to-understand-the-relationship-between-variables/> (accessed on 14 April 2022).
64. Li, A. Human error risk prioritization in crane operations based on CPT and ICWGT. *PLoS ONE* **2024**, *19*, e0297120. [CrossRef] [PubMed]
65. Eglynas, T.; Jusis, M.; Jakovlev, S.; Senulis, A.; Andziulis, A.; Gudas, S. Analysis of the efficiency of shipping containers handling/loading control methods and procedures. *Adv. Mech. Eng.* **2019**, *11*. [CrossRef]
66. Maboudi, M.; Alamouri, A.; De Arriba López, V.; Bajauri, M.S.; Berger, C.; Gerke, M. Drone-Based Container Crane Inspection: Concept, Challenges And Preliminary Results. *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci.* **2021**, *V-1-2021*, 121–128. [CrossRef]
67. Shan, J.; Zhang, M.; Zhang, D.; Zhou, X.; Li, H.; Wang, B. Container Cranes. In *Handbook of Port Machinery*; Springer: Singapore, 2024; pp. 451–678. [CrossRef]
68. Viellechner, A.; Spinler, S. Novel Data Analytics Meets Conventional Container Shipping: Predicting Delays by Comparing Various Machine Learning Algorithms. Available online: <https://hdl.handle.net/10125/63897> (accessed on 12 June 2024).
69. Tian, X.; Yan, R.; Liu, Y.; Wang, S. A smart predict-then-optimize method for targeted and cost-effective maritime transportation. *Transp. Res. Part B Methodol.* **2023**, *172*, 32–52. [CrossRef]
70. Handling Imbalanced Data for Classification—GeeksforGeeks. Available online: <https://www.geeksforgeeks.org/handling-imbalanced-data-for-classification/> (accessed on 12 June 2024).

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.

Article

An Efficient Optimization of the Monte Carlo Tree Search Algorithm for Amazons

Lijun Zhang ^{1,2} , Han Zou ^{1,2} and Yungang Zhu ^{1,2,*}

¹ College of Computer Science and Technology, Jilin University, Changchun 130012, China; zhanglj2121@mails.jlu.edu.cn (L.Z.); zouhan1221@mails.jlu.edu.cn (H.Z.)

² Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, Changchun 130012, China

* Correspondence: zhuyungang@jlu.edu.cn

Abstract: Amazons is a computerized board game with complex positions that are highly challenging for humans. In this paper, we propose an efficient optimization of the Monte Carlo tree search (MCTS) algorithm for Amazons, fusing the ‘Move Groups’ strategy and the ‘Parallel Evaluation’ optimization strategy (MG-PEO). Specifically, we explain the high efficiency of the Move Groups strategy by defining a new criterion: the winning convergence distance. We also highlight the strategy’s potential issue of falling into a local optimum and propose that the Parallel Evaluation mechanism can compensate for this shortcoming. Moreover, We conducted rigorous performance analysis and experiments. Performance analysis results indicate that the MCTS algorithm with the Move Groups strategy can improve the playing ability of the Amazons game by 20–30 times compared to the traditional MCTS algorithm. The Parallel Evaluation optimization further enhances the playing ability of the Amazons game by 2–3 times. Experimental results show that the MCTS algorithm with the MG-PEO strategy achieves a 23% higher game-winning rate on average compared to the traditional MCTS algorithm. Additionally, the MG-PEO Amazons program proposed in this paper won first prize in the Amazons Competition at the 2023 China Collegiate Computer Games Championship & National Computer Games Tournament.

Keywords: Amazons; algorithm optimization; Move Groups; Parallel Evaluation



Citation: Zhang, L.; Zou, H.; Zhu, Y. An Efficient Optimization of the Monte Carlo Tree Search Algorithm for Amazons. *Algorithms* **2024**, *17*, 334. <https://doi.org/10.3390/a17080334>

Academic Editors: Wenxin Li and Haifeng Zhang

Received: 22 April 2024

Revised: 16 July 2024

Accepted: 27 July 2024

Published: 1 August 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Amazons is a two-player board game that was introduced by Walter Zamkaskas of Argentina in 1988 [1,2]. It is the designated game for the Olympia computer game programming competition. Due to its complex gameplay, Amazons is very challenging for humans [3], making it a popular subject for computer game competitions and research. The game rules are as follows: The game is played on a 10×10 square board. Each player controls four identical pieces called “Amazons”. These pieces and arrows follow the same rules as a queen in chess: They can move any number of squares in any direction—horizontal, vertical, or diagonal—but cannot jump over other pieces or arrows. The objective is to restrict the opponent’s movement by shooting arrows until none of the opponent’s four pieces can move. If, on a player’s turn, all four of their pieces are immobilized, that player loses, and the other player wins. Notably, there are no ties in Amazons.

The main challenge of the Amazons game lies in its complexity: Each move offers a vast number of possible choices, typically more than 500 [4,5], and the game can extend for over 80 moves [6,7]. This high level of complexity makes Amazons an excellent experimental sample for testing search strategy algorithms [8,9]. Since its invention, it has garnered significant attention from computer scientists [10] and computer science students [11].

The Monte Carlo method [12] is a computational technique that produces numerical results through repeated random sampling. When integrated with tree search algorithms

and suitable evaluation functions, the Monte Carlo tree search (MCTS) algorithm has yielded impressive results in board games with a high-branching factor (such as Go) [13–15], and in real-time strategy games (such as StarCraft [16]). For instance, in 2016, DeepMind’s AlphaGo program [17] defeated the Korean world champion of Go, Lee Sedol, with a score of 4:1. In 2018, OpenAI’s OpenAI Five program [18] defeated the world champion team in Dota 2 with an overwhelming advantage; both accomplishments are based on the Monte Carlo tree search algorithm.

The ‘Move Groups-Parallel Evaluation’ optimization strategy (MG-PEO) accommodates the rules of the Amazons board game by subdividing a complete move into two parts: Amazon Movement and Arrow Shot. Both Amazon Movement and Arrow Shot are treated as tree nodes, rather than considering a complete move as a single tree node. This approach is known as the Move Groups strategy in the literature [19]. While this strategy has been applied to the Amazons game [20], it still lacks an objective analysis of its effectiveness. Moreover, there is a widespread belief that this method only possesses advantages without any drawbacks. In this paper, we explain the high efficiency of the Move Groups strategy in Amazons by defining a new criterion: the winning convergence distance. We also identify certain shortcomings of this strategy. To address these, we propose using a Parallel Evaluation mechanism to compensate for the limitations of the Move Groups strategy. Our experimental results demonstrate that MG-PEO is an effective fusion optimization strategy.

The main contributions of this paper are as follows:

- We propose an effective optimization of the Monte Carlo tree search algorithm for the Amazons game: the MG-PEO algorithm optimization, which combines the Move Groups strategy with the Parallel Evaluation strategy.
- We provide a performance analysis of the Move Groups strategy in Amazons. By defining a new criterion, the winning convergence distance, we explained the high efficiency of the strategy. Moreover, we also highlight that this strategy has certain shortcomings. The Move Groups strategy involves refining the rules of the Amazons game and using simulations to assign corresponding value scores to additional layer nodes. By utilizing these value scores (e.g., the Upper Confidence Bound Apply to Tree (UCT) strategy [15]), we can achieve effective pruning to save time. However, the addition of extra layer nodes means that more nodes need to be traversed and evaluated in the worst-case scenario, where pruning has not occurred, potentially leading to increased time-consuming overhead. This overhead is incurred as the algorithm strives to find the local optimal solution.
- We propose using a Parallel Evaluation mechanism to address the potential shortcomings of the Move Groups strategy. By parallelizing the evaluation function for the Amazons game, we aim to accelerate the expansion process of the Monte Carlo tree search and minimize the likelihood of the Move Groups strategy falling into a local optimum.

Additionally, we conducted rigorous performance analysis and game experiments. The performance analysis indicates that—compared to the traditional MCTS algorithm—the MCTS algorithm with the Move Groups strategy can enhance the playing ability of Amazons by 20–30 times, while the Parallel Evaluation optimization component can improve the playing ability of Amazons by 2–3 times. Experimental results demonstrate that—compared to the traditional MCTS algorithm—the MCTS algorithm with the MG-PEO strategy achieves an average 23% increase in the game-winning rate and exhibits stronger playing ability.

The remainder of this paper is structured as follows: In Section 2, we present preliminary information about the Amazons game and discuss its relevance to the MCTS algorithm. In Section 3, we provide a detailed description of the MCTS algorithm with the MG-PEO strategy. In Section 4, we offer performance analyses of the Move Groups strategy and Parallel Evaluation strategy, respectively. Section 5 outlines a series of experiments

conducted to verify the efficiency of the MCTS algorithm with the MG-PEO strategy. Finally, conclusions and avenues for future research are summarized in Section 6.

2. Related Work

In this section, we provide an overview of the relevant research that forms the basis of our work. Specifically, we will discuss three key modules in the literature that are relevant to Amazons: historical research on Amazons algorithms, the application of the MCTS algorithm to Amazons, and knowledge about game situation evaluation.

2.1. Amazons Playing Algorithm

The Amazons game has undergone extensive study; its algorithm development can be roughly categorized into three stages. Initially, traditional methods rooted in mathematical theory, such as the minimax search algorithm [21], dominated the field. The second phase saw the emergence of Monte Carlo algorithms, including various variants of the Monte Carlo tree search [19,20,22–24]. In the third stage, deep learning techniques leveraging Amazons game databases came to the forefront [25,26]. Notably, the most representative Amazons program, Invader [27–29], relies on MCTS for prospective exploration, supplemented by neural network training on chess databases, resulting in formidable playing ability in the Amazons game. It can be said that further optimization of Monte Carlo algorithms remains crucial for advancing research on Amazons.

2.2. Monte Carlo Tree Search Algorithm

The development of the Monte Carlo tree search algorithm in Amazons can be roughly divided into two lines of research: algorithm optimization in terms of tree search and algorithm optimization in terms of evaluation functions. In terms of tree search algorithm optimization, researchers have worked on improving search algorithms to improve the performance of Amazons. For example, Guo et al. divided the game into early, middle, and late stages, employing different strategies in each stage [30]. Li et al. applied the UCT algorithm to Amazons [31]; Quan et al. demonstrated the superiority of the UCT algorithm in Amazons [32]. On the other hand, optimizing the evaluation function is also one of the focuses of the research. By optimizing the evaluation function, the current game can be evaluated more accurately and the search algorithm can be guided to make more informed decisions. Lieberum et al. dedicate their research to this aspect [33,34], focusing mainly on improving hyperparameter tuning and designing new evaluation functions. Overall, the Monte Carlo tree search algorithm cannot be improved without any one aspect, and in this paper, our proposed fusion of the Move Groups strategy and the Parallel Evaluation strategy represents a comprehensive consideration of the two historical research focuses. This fusion of the Monte Carlo tree search algorithm is cutting-edge and effective.

2.3. Evaluation Function

At present, almost all search algorithms are faced with a problem: situation evaluation. Good situation evaluation is very important in the Amazons game and can even determine the quality of the game program [33].

Similar to many board games, Amazons can be roughly divided into three stages: opening, middle, and ending, with distinct strategies for each stage. The main indicators of the Amazons evaluation function include territory, position, and mobility, and the importance of these three characteristics varies across different stages. Based on these ideas, Guo et al. [30] proposed a phased evaluation function, as shown in Equation (1):

$$Value = k_1 * t_1 + k_2 * t_2 + k_3 * p_1 + k_4 * p_2 + k_5 * mobility \quad (1)$$

where t_1 and t_2 are territory metric values, which, respectively, represent the evaluation of each side's control over space, based on the QueenMove Method and KingMove Method. p_1 and p_2 are position metric values, reflecting the strategic position of each side's pieces, also based on the QueenMove and KingMove Methods relative to the space. *mobility*

is the flexibility evaluation value of both Amazon pieces. k_1, k_2, k_3, k_4 , and k_5 are the corresponding weight values. How these values are calculated is described below.

According to the literature [2], calculating territory and position values involves two types of methods: QueenMove and KingMove. QueenMove means that a move can be made as long as there are no obstacles to be crossed in any of the eight directions (the obstacles include pieces and arrows), while KingMove means that the distance of a single move is one, i.e., only one move can be made to a neighboring square. The QueenMove value represents the minimum number of moves a player's four pieces need to reach a given space (i.e., no pieces or arrows); the KingMove value represents the minimum number of moves a player's four pieces need to make to reach a given space. As shown in Figure 1 on the left side, value 2 in the upper-left corner of the upper-left (A, 1) square indicates that the white piece needs at least two moves to reach this space via the QueenMove method, and value 4 in the lower-right corner indicates that the black piece needs four moves. Similarly, values 2 and 7 in the same square in Figure 1 on the right side indicate the situation when the KingMove method is used.

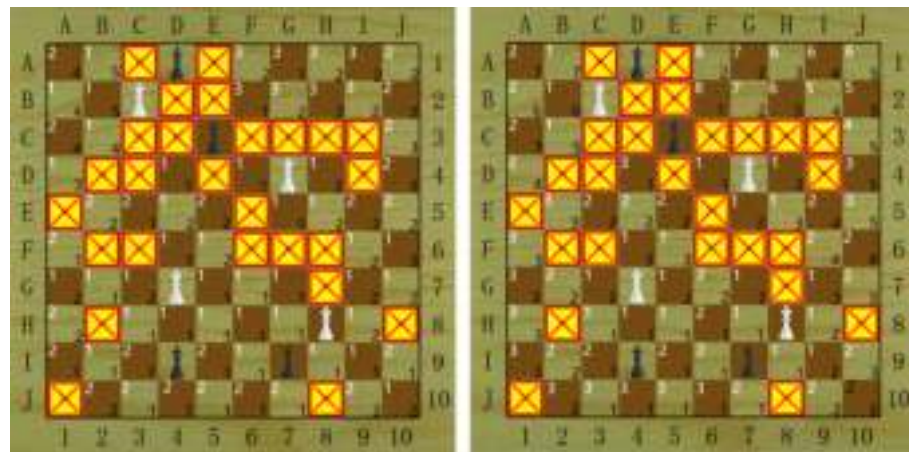


Figure 1. Calculation results of the QueenMove and KingMove methods. The left is the result of the KingMove calculation, and the right is the result of the QueenMove calculation.

Accordingly, we can use the following calculation formula for all the indicators:

t_1 represents the difference in the control of the territory using the QueenMove method.

t_2 represents the difference in the control of the territory using the KingMove method. As shown in Equations (2) and (3):

$$t_i = \sum v(B_i, W_i) \quad (2)$$

$$v(B_i, W_i) = \begin{cases} 0, B_i(A) = W_i(A) = \infty \\ k, B_i(A) = W_i(A) \neq \infty \\ 1, B_i(A) < W_i(A) \\ -1, B_i(A) > W_i(A) \end{cases} \quad (3)$$

where B represents black pieces, W represents white pieces, with the subscript 1 indicating the QueenMove method and 2 indicating the KingMove method. A represents empty squares. k represents the preponderance of the leading square, and the value range is greater than -1 and less than 1 . When the player with the first move is white, k is greater than 0 , otherwise, it is less than 0 .

p_1 represents the positions of the two pieces on control space A using the QueenMove method. p_2 represents the positions of the two pieces on control space A using the KingMove method. As shown in Equations (4) and (5):

$$p_1 = 2\Sigma \left(2^{-B_1(A)} - 2^{-W_1(A)} \right) \quad (4)$$

$$p_2 = \sum_A \min(1, \max(-1, B_2(A) - W_2(A)/6)) \quad (5)$$

mobility means the flexibility of the pieces. As shown in Equations (6) and (7):

$$mobility = \sum_{w \in \text{white piece}} F(w) - \sum_{b \in \text{black piece}} F(b) \quad (6)$$

$$F(a) = \begin{cases} \text{surrounding spaces,} & a \text{ is the space} \\ 0, & a \text{ is the obstacle} \\ \sum_{s \in S_a} \frac{F_{space}(s)}{D(a, s)}, & a \text{ is an Amazon piece} \end{cases} \quad (7)$$

where F denotes flexibility, S_a denotes the set of spaces that can be reached by a piece using the QueenMove method, and $D(a, s)$ is the minimum number of moves by a to s using the KingMove method.

We retained the hyper-parameterization settings according to the literature [2]; detailed information about each assessment indicator is shown in Table 1.

Table 1. Weight coefficient table.

	Evaluation Metrics	Territory		Position		Mobility
	Evaluation Factor Weight Coefficient	t_1 k_1	t_2 k_2	p_1 k_3	p_2 k_4	<i>mobility</i> k_5
Stages	Opening	0.14	0.37	0.13	0.13	0.20
	Middle	0.30	0.25	0.20	0.20	0.05
	Ending	0.80	0.10	0.05	0.05	0.00

Table 1 shows that t_1 gradually increases in importance as the game progresses, starting smaller in the opening, becoming larger in the middle, and reaching its peak in the endgame. This is because as the game progresses, board control becomes more crucial to victory or defeat. t_2 is extremely crucial in the opening phase to ensure the safety of neighboring regions and territorial boundaries, but its role decreases as the game progresses, and it can be almost ignored in the endgame. It reflects the urgency of the initial layout defense and territorial competition. p_1 and p_2 denote piece positions, which mainly play roles in the opening phase when a balanced distribution of pieces is crucial to the subsequent strategic deployment; in the endgame, the pattern of the game has already been determined, and the distribution of pieces is no longer a key point; *mobility* denotes flexibility, which is the most significant in the early stage of the game, as blocking the pieces can be seriously disadvantageous; as the game progresses and the initial layout is established, the distribution of pieces becomes increasingly critical. As the game progresses and the layout is completed, the impact of piece mobility on the game's situation is weakened, so its weight is reduced to negligible in the middle and late game. This reflects the need for strategic flexibility in the early stages of freedom of movement.

3. Research Methodology

In this section, we first describe the overall framework of the MCTS algorithm enhanced with the MG-PEO strategy. We then elaborate on the optimization details, focusing on two aspects: Move Groups and Parallel Evaluation.

3.1. Overall Framework of Algorithm

We propose an MG-PEO algorithm optimization strategy based on the Monte Carlo tree search [22]. As shown in Figure 2, the optimized Monte Carlo tree search algorithm involves iteratively constructing a search tree, creating a new tree each round until the time constraint is reached or the game ends. At this point, the search stops and returns the best-performing root operation (a combination of actions from certain nodes in the second and third layers).

Specifically, the following four steps are performed for each search iteration:

- (1) Selection: Starting from the root node, a certain strategy is applied to recursively go down through the tree for expandable node selection. A node is expandable if it represents a non-terminal state and has unvisited children.
- (2) Expansion: Expand the tree by adding one (or more) child node(s), depending on the available operations.
- (3) Simulation: Run a simulation from a new node to produce results according to the default strategy.
- (4) Backpropagation: Back up the simulation results through the selected nodes to update their statistics.

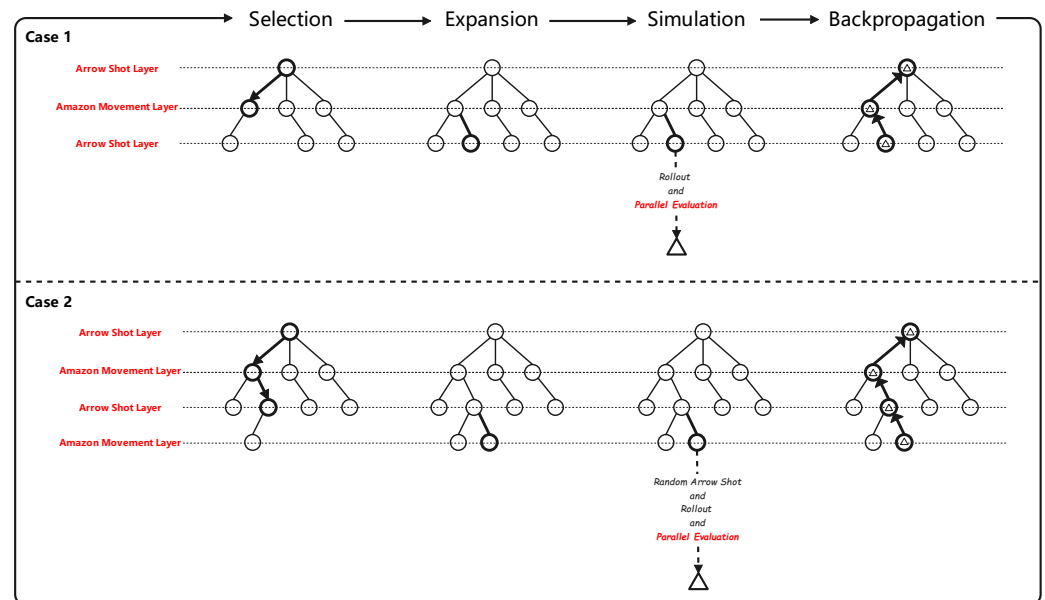


Figure 2. MCTS plus MG-PEO general framework.

As seen in Figure 2, due to the use of the Move Groups strategy, the algorithm operates under two scenarios during the simulation phase: if the current node is an Arrow Shot layer node, it executes Rollout and Parallel Evaluation; if the current node is an Amazon Movement layer node, it performs Random Arrow Shot, Rollout, and Parallel Evaluation.

The pseudo-code for the overall framework is described in Algorithm 1.

Algorithm 1: General Algorithm Framework

Input: s_0 (The current game State)

Output: Optimal decision

```

1: function MCTSSEARCH ( $s_0$ )
2:   create root node  $v_0$  with state  $s_0$ 
3:   while within the computational budget do
4:      $v_1 \leftarrow$  Selection ( $v_0$ )
5:      $v_2 \leftarrow$  Expansion ( $v_1, s_0$ )
6:      $\Delta \leftarrow$  Simulation ( $v_2, s_0$ )

```

```

7:      Backup ( $v_2, \Delta$ )
8:  return BESTCHILD( $v_0$ )

```

3.2. Move Groups Strategy in Algorithm

The core idea of the Move Groups strategy is as follows: according to the rules, a node in the original search tree (Amazon Movement and Arrow Shot combined) is decomposed into two nodes (representing Amazon Movement and Arrow Shot, respectively). By utilizing the simulation phase of MCTS to assign value scores to an additional layer (the newly added layer), we can achieve the goals of pruning and saving time, based on the value scores of the extra layer (e.g., using the UCT strategy).

Specifically, when an extra layer node has not yet expanded its child nodes, its value score is initially infinite. Therefore, each extra layer node will first expand its child nodes once and employ a Shallow Rollout strategy to obtain an initial value score. Subsequently, we can select the extra layer node with the highest value score for further expansion and Rollout, until a particular extra layer node's child nodes are fully generated and the node is selected again. At this point, we can then derive the optimal decision from the child nodes, which is the core of early pruning. It should be noted that due to the dynamic MCTS backpropagation mechanism, the value scores of the extra layer nodes will dynamically change as their child nodes expand. Therefore, early pruning may also occur at the second, third, or other extra layer nodes. For details on the computation and updating of the value scores, please refer to Section 3.3. Since the complete step is a combination of two steps, namely, Amazon Movement and Arrow Shot, and the extra layer represents only one step, the total number of nodes in the extra layer must be less than the total number of nodes, thus enhancing the performance of early pruning in the Monte Carlo tree search algorithm.

However, the Move Groups strategy also increases the number of total tree nodes (for a complete one-step operation) due to the introduction of the extra layer, and if all nodes are expanded without pruning in advance, the extra overhead (expansion and evaluation) incurred by the nodes in the extra layer will negatively affect the algorithm, which will also lead to the fact that—in the worst-case scenario—the addition of the Move Groups strategy to the MCTS will perform slightly worse than the MCTS algorithm without the Move Groups policy. A complete theoretical analysis of the Move Groups policy is detailed in Section 4.1.

3.3. Parallel Evaluation Strategy in Algorithm

The core idea of the Parallel Evaluation strategy is that the time-consuming evaluation part of the MCTS simulation is processed in parallel optimization. It has been shown in the literature [35] that accelerating this labor-intensive evaluation function allows the Monte Carlo Tree Search algorithm to be iterated more frequently. This metric is very beneficial in compensating for the fact that the Move Groups strategy might fall into a local optimum (worst case); specifically, speeding up the evaluation function enables the Monte Carlo Tree Search to scale as quickly as possible, minimizing the impact of the worst-case scenario associated with the Move Groups strategy. The pseudo-code for the Parallel Evaluation part is as shown in Algorithm 2.

Algorithm 2: SituationEvaluation

Input: s_0 (The current Game State)

Output: Situation Evaluation Value

```

1: function SituationEvaluation( $s_0$ )
2:    $t_1, t_2, p_1, p_2, mobility \leftarrow$  ParallelEvaluationMetrics ( $s_0$ )
3:    $Value = k_1 * t_1 + k_2 * t_2 + k_3 * p_1 + k_4 * p_2 + k_5 * mobility$ 
4:   return Value

```

Nodes undergo evaluation, and the resulting value scores guide node selection. Traditional Monte Carlo tree search algorithms typically choose the child node with the highest

node score (winning rate) based on the current game state [36]. However, this selection method is not ideal, as it tends to prioritize immediate gains and can lead to local optima. To address this issue, this paper incorporates the “Shallow Rollout strategy” [37] and the “tree-based upper confidence interval” algorithm [19] to refine node selection. Specifically, the utilization value derived solely from decision-level nodes may be inaccurate. To enhance precision, a Shallow Rollout strategy is employed, simulating scenarios closer to the endgame. Additionally, the algorithm considers the magnitude of a node’s exploratory value, which is elevated for nodes exhibiting high uncertainty and numerous future possibilities. In summary, the approach assesses both partial equilibrium utilization value and exploration value to facilitate more informed decision-making.

1. “Shallow Rollout Strategy”: This means that the utility value is not calculated, not only based on the next move but also on the position of the game after randomly simulating the move several times. In this algorithm, we set the number of Rollouts to be two, and the depths to be 4 and 5. Specifically, after expanding the complete decision node (Amazon Movement plus Arrow Shot), we perform two Rollouts for the current situation. In each Rollout, we simulate two steps: the opponent’s next decision (Amazon Movement plus Arrow Shot) and our subsequent decision (Amazon Movement plus Arrow Shot), to obtain a more forward-looking utility value. See pseudo-code Algorithm 3 for details. For the analysis of the relevant parameters, see Appendix A.
2. “Tree-based Upper Confidence Interval”: As shown in Equation (8), the calculation formula of UCT consists of two components. The first term, X_i , represents the utility value of the information in the constructed search tree, and the second term represents the exploration value of the unvisited nodes. The constant C is used to adjust the proportion of the algorithm’s trade-off between the two values. Note that the value of the root formula is infinite because of the presence of unreachable child nodes, which ensures that unreachable child nodes under the current node will be visited at least once.

$$UCT = X_i + C \sqrt{\frac{\ln N}{n_i}} \quad (8)$$

where X_i is the winning rate of the child node. N is the number of times the parent node is accessed. n_i is the number of times the child node is accessed. C is a constant ($\frac{1}{\sqrt{2}}$). Kocsis and Szepesvári [36] showed that the value satisfies the Hoeffding inequality with rewards in the range of $[0, 1]$.

Here, we can provide the complete pseudo-code as shown in Algorithm 3.

Algorithm 3: MCTS with the MG-PEO strategy

```

1: function Selection( $v$ )
2:   while  $v$  is nonTerminal do
3:     if  $v$  is not fully expanded then
4:       return  $v$ 
5:     else
6:        $v \leftarrow \text{BESTCHILD}(v)$ 
7:   return  $v$ 
8: function Expansion( $v, s_0$ )
9:   if  $v$  is an Amazon Movement Node then
10:     $ch \leftarrow$  randomly create Arrow Shot Node
11:   else if  $v$  is an Arrow Shot Node
12:     $ch \leftarrow$  randomly create the Amazon Movement Node
13:    $s_0 \leftarrow$  Update the state of the board
14:   return  $ch$ 
15: function Simulation( $v, s_0$ )

```

```

16:   $S \leftarrow s_0$ 
17:  /* depth = 4 and 5 (Independent order twice)
18:   $S \leftarrow \text{Rollout}(S, \text{depth})$ 
19:   $\Delta \leftarrow \text{SituationEvaluation}(S)$ 
20:  return  $\Delta$ 
21: function Rollout( $S, \text{depth}$ )
22:   repeat
23:     if is Terminal ( $S$ ) then
24:       pass
25:     if  $v$  is the Amazon Movement Node then
26:        $\text{arrow} \leftarrow \text{RandomArrow}(S)$ 
27:        $S \leftarrow \text{doAction}(v, \text{arrow})$ 
28:        $\text{depth} \leftarrow \text{depth} - 1$ 
29:     else if  $v$  is the Arrow Shot Node then
30:        $\text{move} \leftarrow \text{parent of } v$ 
31:        $S \leftarrow \text{doAction}(\text{move}, v)$ 
32:        $\text{depth} \leftarrow \text{depth} - 1$ 
33:   until  $\text{depth} = 0$  or gameover
34:   return  $S$ 
35: function Backup( $v, \Delta$ )
36:   $N(v) \leftarrow N(v) + 1$ 
37:   $Q(v) \leftarrow Q(v) + \Delta(v, \text{parent of } v)$ 
38:   $v \leftarrow \text{parent of } v$ 
39: function BESTCHILD( $v$ )
40:  return  $\text{argmax}_{v' \in \text{children of } v} \frac{Q(v')}{N(v')} + c \sqrt{\frac{2 \ln N(v)}{N(v')}}$ 

```

4. MG-PEO Performance Analysis

In this section, we design two modules to analyze the MG-PEO strategy. First, we analyze the performance of the Move Groups strategy in Amazons. Second, we conduct an in-depth discussion on the performance improvement of the Parallel Evaluation mechanism in Amazons.

4.1. Performance of the Move Groups Strategy in Amazons

The effectiveness of a Monte Carlo tree search algorithm based on the law of large numbers depends largely on the number of situations it can search. The more situations the algorithm can search for, the easier it will be to find “tricks” that are usually hard to find and, thus, beat the opponent. In other words, the more computational the power, the more effective the Monte Carlo algorithm. Therefore, in order to objectively show the influence of the Move Groups strategy on the actual performance of the MCTS algorithm, we provide a new definition (winning convergence distance) to evaluate the performance of the algorithm.

Definition 1. Winning convergence distance. This calculates the number of tree nodes expanded by the search tree, both in depth and breadth, from the current state (root node) to the winning state (game over). In general, the fewer new nodes the Monte Carlo tree needs to expand to achieve victory as soon as possible, the shorter the winning convergence distance.

To visualize the winning convergence distance, we use the opening first step as an example. Since the number of nodes that can be generated from the first step to the winning state is immeasurable, we use the local optimal solution of the first step instead of the winning state, as shown in Figure 3.

Based on this, we can plot the Monte Carlo search tree with the Move Groups strategy added and the Monte Carlo search tree without the Move Groups strategy added,

and compute the best-case and worst-case expanded node numbers, as shown in Figure 4 and Table 2.



Figure 3. The local optimal solution assumed in the first step of the opening. From left to right: initial state, after Amazon Movement, and after Arrow Shot.

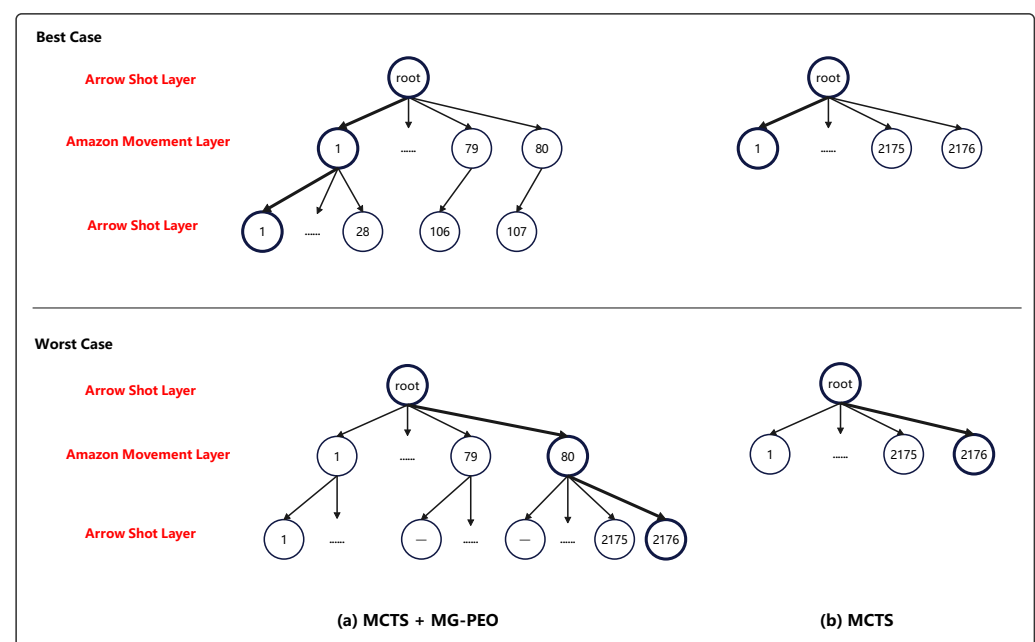


Figure 4. Plot of the winning convergence distance calculation. The number in the figure refers to the order in which the node is expanded at a certain layer.

Table 2. Winning convergence distance of two algorithms (unit: nodes).

Algorithm	Best Case	Worst Case
MCTS + MG-PEO	187	2256
MCTS	2176	2176

Red results indicate the best result in the same column.

In Figure 4 and Table 2, for the first step, the Monte Carlo tree search algorithm with the Move Groups strategy added needs to expand 80 Amazon Movement layer nodes first. Since the value score of an Amazon Movement layer node is infinite when it has not yet expanded its child nodes, each Amazon Movement layer node needs to further expand an Arrow Shot layer node to obtain an initial value score. Subsequently, we can select the Amazon Movement layer node with the highest value score for the next child node expansion. In the best case, the first Amazon Movement layer node has the highest value score. During the expansion of its subsequent Arrow Shot layer nodes, the value score of the parent node remains at the maximum observed across all Amazon Movement layer nodes; the other nodes in the Amazon Movement layer do not carry out the expansion of

the next layer's nodes, which is a reflection of early pruning. Finally, the number of nodes expanded is 187. In the worst case, all possible nodes are fully expanded and traversed, resulting in a total of 2256 nodes being expanded.

The Monte Carlo tree search algorithm without the inclusion of the Move Groups strategy needs to select the node with the largest value score as the final decision after generating all possible nodes, regardless of the best or worst-case scenarios; resulting in a total of 2176 nodes being expanded.

In summary, we can conclude the following: in the best case, the winning convergence distance of the MCTS with the Move Groups strategy is significantly shorter than that of the traditional MCTS, while in the worst case, the winning convergence distance of the MCTS with the Move Groups strategy is slightly weaker than that of the traditional MCTS. This indicates that the Move Groups strategy has strong pruning and time-saving abilities, but at the same time, there are some shortcomings.

4.2. Performance of Parallel Evaluation Strategy

In this subsection, we delve into the number of threads required for the Parallel Evaluation mechanism, as well as the speedup ratio that multithreading can bring. Since the time consumption of the evaluation function mainly comes from the time from calculating each evaluation metric, the Parallel Evaluation part is essentially a multi-threaded calculation of evaluation metrics. The pseudo-code for the Parallel Evaluation metric function is shown in Algorithm 4.

Algorithm 4: Parallel Evaluation Metrics

Input: s_0 (The current Game State)

Output: $t_1, t_2, p_1, p_2, mobility$

```

1: function ParallelEvaluationMetrics( $s_0$ )
2:   /* The calculation of Territory( $t_1, t_2$ ) and Position( $p_1, p_2$ ) metrics uses the same
      independent function four times, based on the King Move and QueenMove
      methods, which are calculated once for black and once for white.
3:   /* We optimize its external calls in parallel.
4:   Module1 = lambda (range)
5:     for the  $index$  in the range:
6:        $color \leftarrow index/2$ 
7:        $type \leftarrow index\%2$ 
8:       CalTP( $color, type$ )
9:   parallel_for(block_range(0,2),Module1)

10:  /* The Mobility( $mobility$ ) metric is calculated one function at a time, and the
      internal logic is to process each board position ( $GridSize = 100$ ) independently.
11:  /* We optimize its inner loop processing logic in parallel.
12:  Module2 = lambda (range)
13:    for the  $index$  in the range:
14:       $color \leftarrow index/2$ 
15:       $type \leftarrow index\%2$ 
16:      CalEveryBoardPosition()
17:  parallel_for(block_range(0,GridSize),Module2)

```

A. Discussion on the number of threads: Since the main metrics of the evaluation function include territory, position, and maneuverability metrics, values of the territory and position metrics can be calculated using a single function; however, as the calculations of these two metrics are related to the colors of the pieces and the calculation method (KingMove method and QueenMove method), four independent calculations are needed for the values of the territory and position metrics. The mobility indicator is calculated by a function that requires one independent calculation. The algorithm is optimized in

parallel for the territory and position indicators, which theoretically require a total of four threads. For the maneuverability indicator, ten threads are theoretically needed for internal calculations related to the size of the board.

B. Exploration of the acceleration ratio that multi-threading can bring: In order to accurately explore the acceleration ratio, we counted the time consumed by the computation function of each evaluation metric, and the data are shown in Table 3.

Table 3. Module execution time and speed-up ratio.

Module	Serial Execution Time	Parallel Execution Time	Speed-Up Ratio
Module1	69 us	29 us	2.379
Mudule2	45 us	23 us	1.956
Whole Evaluation Function	114 us	54 us	2.111

As shown in Table 3, the calculation time for the optimized territory and position metrics achieved a 2.379-fold speedup, while the calculation time for the optimized mobility metric achieved a 1.956-fold speedup. The overall speedup ratio for the evaluation part was 2.111. In section A, we discussed the number of threads and noted that the theoretical speedup of parallelization should correspond to the number of threads. However, the time consumption of each module before and after optimization differed somewhat from the discussion in section A. We consider that the overhead of third-party parallel libraries[38] has consumed some computing resource performance. See Appendix B for details on the CPU usage analysis. Nevertheless, the parallel optimization mechanism reduced the time required to process the same tasks by 2.111 times, resulting in a significant performance improvement in practical applications.

5. Experiment and Results

In this section, in order to verify the effectiveness of the MG-PEO algorithm optimization strategy proposed in this paper, three experiments were designed: the first compares the relevant performance metrics of the MCTS algorithm with the MG-PEO strategy and the traditional MCTS algorithm, and a significance test is carried out on the comparison results. The second experiment focuses on assessing the magnitude of the effect of each optimization module of MG-PEO on MCTS, i.e., the ablation experiment. The third experiment is a generalized experimental analysis of the MG-PEO.

In addition, it is worth mentioning that the MG-PEO Amazons program proposed in this paper won first prize in the Amazons Competition at the 2023 China Collegiate Computer Games Championship & National Computer Games Tournament.

5.1. Experimental Environment Information

In order to ensure the reliability of the experimental results, we conducted all the experiments in a unified environment, and detailed information on the experimental environment settings is shown in Table 4.

Table 4. Experimental environment information.

Hardware and Constraints	Details
GPU	GeForce RTX 3070
CPU	Intel Core i7-11800H
Decision-Making time	5 s
Number of matches played by the same opponent	2
Number of sets per match	3
Experimental Matching Platform	SAU GAME Platform
Visual Studio Runtime Mode	[Release x64]

Against the same opponent, a total of six sets will be played, with three sets as the first player and three as the second player.

5.2. Quantitative and Qualitative Experiments

5.2.1. Winning Percentage Statistics and Analysis

In order to objectively and accurately assess the effectiveness of the MG-PEO algorithm optimization strategy, we selected the MCTS algorithm without Move-Groups optimization and without Parallel Evaluation optimization for comparison, and we played a total of 100 matches under the same conditions, with each match divided into 3 games each (first hand and second hand). Specifically, we chose 33 players who participated in a computer gaming competition and 17 players who open-sourced their rankings (ladder list) on the Botzone platform (<https://www.botzone.org.cn/>, accessed on 2 March 2024), totaling 50 opponents. Due to the requirement for third-party libraries, we locally exported the exe programs for all players and then conducted the games against each other on the SAU platform (http://computergames.caii.cn/platform/SAU_Game_Platform_2.1.0_r3.rar, accessed on 20 March 2024), and the results of the games are shown in Table 5. Detailed information on all the games can be found in Appendix C.

Table 5. The winning rate of the two algorithms.

Algorithm	Winning Percentage
MCTS	74%
MCTS + MG-PEO	97%

Red results indicate the best result in the same column.

In Table 5, the MCTS algorithm has a winning rate of 74% in Amazons Duel on the SAU matchmaking platform. In the same test environment, the MCTS algorithm with the MG-PEO strategy shows higher stability, and its winning rate in the Amazons Match on the SAU Matchmaking Platform reaches 97%. In summary, the MCTS algorithm with the MG-PEO strategy has more comprehensive and in-depth optimization when considering and utilizing strategies than the MCTS algorithm and, thus, shows a higher winning rate in this test scenario.

It is worth noting that—by observing the matches of the two algorithms—we found that the optimized MCTS algorithm often makes more accurate decisions than the unoptimized MCTS algorithm in some critical rounds. These key rounds often involve crucial positions for “encircle territory” or “block the opponent” strategies. Additionally, during the matches, we noticed that the optimized MCTS algorithm has a clear advantage over some traditional algorithms (such as Minimax and traditional MCTS). However, its performance is slightly lacking when competing against neural network programs trained with a large amount of game data. Nevertheless, this provides a direction for further improving the algorithm’s performance in the next step.

5.2.2. Winning Convergence Distance Visualization and Analysis

In this sub-subsection, we count the number of match point rounds in which the algorithm’s winning percentage assessment reaches 90% or more (the threshold for a solid win) in 100 games played by the two algorithms on the SAU platform, as a way to visualize the definition of the distance of convergence of a win, as well as to compare the strengths and weaknesses of the playing abilities of the two algorithms of the Amazons game. Here, we generally consider the outcome of a game to be clear when the algorithm evaluates a winning rate of 90% or more (barring low-level errors by the dominant side); such moments are considered to be match points and we believe that the earlier this occurs, the stronger the algorithms are. The average resulting data for the number of match point rounds for the two algorithms are shown in Table 6. See Appendix C for complete data:

Table 6. The average number of match point rounds in 100 matches where the win rate evaluation values of two algorithms exceed 90%.

Algorithm	Mean Value
MCTS	42.57
MCTS + MG-PEO	39.68

If the player loses, the match point rounds are taken as the maximum number of rounds, i.e., 47. Additionally, Red results indicate the best result in the same column.

As can be seen from Table 6, the average number of match point rounds required for the MCTS algorithm to reach the stable winning threshold (evaluation value over 90%) is 42.57 rounds. On the other hand, the MCTS algorithm with the MG-PEO strategy outperforms MCTS in terms of reaching the same stable winning threshold with an average of 39.68 rounds.

This indicates that the MCTS algorithm with the MG-PEO strategy converges faster than MCTS. It requires fewer average rounds than the traditional MCTS algorithm. The MCTS algorithm with the MG-PEO strategy uses the Move Groups strategy and the Parallel Evaluation strategy to optimize the strategy selection process of the MCTS algorithm so that it can find the best search paths in a shorter period of time and reach a stable winning state with a winning rate of more than 90% in advance.

In summary, the MCTS algorithm with the MG-PEO strategy has a shorter winning convergence distance than the traditional MCTS algorithm. It requires fewer rounds to achieve a higher winning rate and, thus, demonstrates superior playing ability.

5.2.3. Significance Test

In this subsection, we perform a significance test on the match point round count data, as counted in Section 5.2.2, to determine whether there is a significant difference between them. We conduct the tests at a significance level of 0.05. The complete match point round count data are detailed in Appendix C. First, we use the Shapiro–Wilk test to assume that both data sets obey a normal distribution. However, neither dataset conforms to a normal distribution. Based on this result, we use the Mann–Whitney U test to assume that there is no significant difference between the two groups of data. We then compare the distribution characteristics of the match point round data of the two algorithms. The detailed results are shown in Table 7.

Table 7. Significance of the test results.

Match Point Rounds Data	<i>p</i> -Value for Shapiro–Wilk Test (P_1)	<i>p</i> -Value for Mann–Whitney U Test (P_2)
MCTS	0.0002	1.16×10^{-8}
MCTS + MG-PEO	0.0013	

As a result of the Shapiro–Wilk test, both algorithms tested showed a P_1 value of less than 0.05 for the data on the number of match point rounds, leading us to reject the null hypothesis that the data for MCTS+MG-PEO and the data for MCTS both conform to a normal distribution. The results of the Mann–Whitney U test indicate that the P_2 value for the data on the number of match point rounds between the two algorithms is significantly smaller than 0.05, so we reject the original hypothesis. That is, there is a significant difference in the average number of winning steps between the two algorithms. Specifically, the average number of winning moves required by MCTS with the MG-PEO strategy is significantly less than that required by traditional MCTS against 100 players, suggesting that MCTS with the MG-PEO strategy outperforms traditional MCTS in terms of playing ability in Amazons.

5.3. Ablation Experiment

In this subsection, to evaluate the effectiveness of the Move Groups strategy and the Parallel Evaluation strategy in Amazons, we design two experiments: one that only employs the Move Groups strategy without using the Parallel Evaluation strategy, and another that only employs the Parallel Evaluation strategy without using the Move Groups strategy. The experimental results are shown in Table 8. On the one hand, the Move Groups strategy focuses on MCTS iterations within a limited time on the sub-nodes of more valuable additional-layer nodes by leveraging their value scores, thus avoiding the expansion of unimportant nodes. While the Move Groups strategy exhibits strong pruning capabilities, the value scores of additional-layer nodes dynamically change with the expansion of their sub-nodes due to the MCTS dynamic backpropagation update mechanism. Consequently, all nodes in the additional layer may be fully expanded and traversed multiple times (pruning has not occurred), potentially resulting in increased time spent compared to MCTS algorithms without the Move Groups strategy. On the other hand, because the proportion of the situation evaluation is limited, using only Parallel Evaluation optimization without adopting the Move Groups strategy offers limited performance improvement. Therefore, the combined optimization of employing both the Move Groups strategy and the Parallel Evaluation is effective and necessary.

Table 8. Ablation experiment analysis of the Move Groups strategy and Parallel Evaluation strategy.

Algorithm	Winning Percentage	Number of MCTS Iterations within 1s
MCTS	74%	2176
MCTS + PEO	89%	10,026
MCTS + MG	83%	5663
MCTS + MG-PEO	97%	45,536

The “Number of MCTS Iterations Within 1s” refers to the number of Monte Carlo tree search iterations conducted within one second. Under the Move Groups strategy, each MCTS iteration results in either an Amazon Movement node or an Arrow Shot node. Without using the Move Groups strategy, each MCTS iteration results in a complete decision node (Amazon Movement plus Arrow Shot). Additionally, Red results indicate the best result in the same column.

5.4. Generalization Analysis

In this subsection, we conducted a general analysis of the Move Groups strategy and the Parallel Evaluation strategy, discussing in depth their transferability to other board games.

1. Move Groups strategy: In some games, the branching factor can be very large, but many moves are similar. The Move Groups strategy creates an additional decision layer where all possible actions are grouped together, allowing us to handle the large branching factor of the game more efficiently. This strategy is beneficial for other games, such as Go [39] and the multi-armed bandit game with Move Groups [40]. We believe that this strategy is still worth exploring and investigating.
2. Parallel Evaluation strategy: The Parallel Evaluation strategy can provide effective acceleration, whether for simple rule-based games like Gomoku, more complex games like Chess and Go, or even modern innovative board games. Its core advantage lies in its ability to efficiently explore the decision tree, which is a common requirement in various board games.

6. Conclusions and Future Works

In this paper, we propose an efficient optimization of the Monte Carlo Tree Search algorithm for the Amazons game, which refines the Monte Carlo Tree Search capability through the Move Groups strategy. The Parallel Evaluation mechanism addresses the risk of the Move Groups strategy falling into a local optimum (these components are

complementary to each other). The experimental results validate that the method proposed in this paper significantly improves the playing ability of the Amazons game.

Future research efforts are also included in our plans. First, to maximize the timesaving potential of the Move Groups strategy pruning, we hope to consider using more intelligent approaches (such as heuristic functions) to guide the pruning behavior of the Move Groups strategy. Secondly, the series of experiments in this paper demonstrate the powerful performance improvement capability of parallel optimization, so we plan to consider more parallel possibilities and more efficient parallel strategies in the future. Finally, we also hope that this algorithm can be further extended to solve more gameplaying problems.

Author Contributions: All authors contributed to the study’s conception and design. Methodology, software, validation, investigation, data analysis, model verification, resources, data curation, visualization, writing—editing, and review: L.Z. and H.Z. Supervision, project administration, funding acquisition: Y.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Part of the competition code used in this study is the publicly available open source code, which can be found at the following website: <https://www.botzone.org.cn/>, accessed on 2 March 2024.

Conflicts of Interest: The authors declare that they have no financial interests or personal relationships relevant to the content of this article’s content.

Appendix A

In Appendix A, we discuss and analyze the effect of the depth of Rollout in the simulation phase on the algorithm’s ability to play Amazons, and use the winning convergence distance metric (see Section 4.2 for details) to evaluate the playing ability. Shallow Rollout strategies can consider simulated positions closer to the endgame and more forward-looking games to compute more accurate position evaluation values, which can be used to better guide the node selection for MCTS. Therefore, it is crucial to balance the depth of Rollout; too shallow a depth will make the utilization value (winning rate) in the position evaluation formula less credible, while too deep a depth will lead to a single simulation iteration process that is too long, and the number of nodes expanded out in a finite period of time will become smaller, reducing the accuracy of the Monte Carlo tree search decision. We adjusted the Rollout depth value from 2 to 10 and found that the optimal value of Rollout depth is 4 or 5, as shown in Table A1.

Table A1. Parametric analysis of the Rollout depth.

Rollout Depth	1	2	3	4	5	6	7–10	4 and 5
Match Point Rounds	40.25	40.14	40.0	39.9	39.84	40.04	40.1	39.68

Red results indicate the best result in the same column.

In addition, we found in our experiments that Rollout twice (choosing 4 and 5 in-depth) works better than Rollout once (choosing only 4 or 5), i.e., the number of rounds at match points is smaller, the distance of convergence for winning is shorter, and the algorithm is more capable of playing Amazons.

Appendix B

Appendix B shows the detailed CPU usage of the algorithm with and without the Parallel Evaluation strategy. The monitoring environment is the performance profiler in Microsoft Visual Studio 2019, with the algorithm making only one decision, which takes 20 s. As shown in Figure A1, the top part of the image represents the CPU utilization, and the fluctuating part of the CPU utilization values corresponds to the algorithm’s decision-making phase, while the remaining part corresponds to the input and output idle phases.

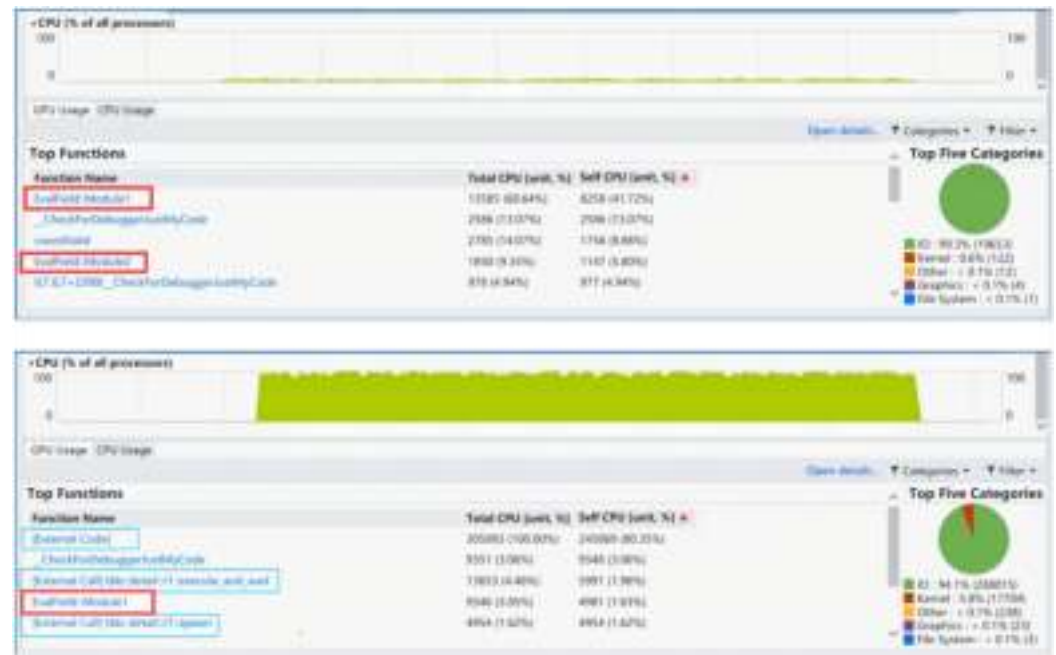


Figure A1. CPU utilization and top 5 modules occupying CPU resources under [Debug x64] condition. The upper part shows the monitoring results of the algorithm executed sequentially, while the lower part shows the monitoring results of the algorithm executed in parallel.

From Figure A1, it is evident that the CPU utilization of the algorithm program in parallel execution is significantly higher than in sequential execution. When the algorithm program executes sequentially, both red-framed modules of the evaluation function are among the top 5 modules occupying the most CPU resources. This indicates that parallel optimization of the evaluation part is very necessary. During parallel execution of the algorithm program, the blue-framed third-party parallel library occupies a substantial amount of CPU resources. This indirectly indicates that the performance improvement after parallelization does not achieve the theoretical speedup due to significant resource consumption by external code and external calls.

Appendix C

Appendix C shows the complete game information, as well as the match point round count, as shown in Tables A2 and A3.

Table A2. Complete game information (MCTS plus MG-PEO).

ID	First Player	Second Player	Result	Match Point Rounds	Winner
1	Amazons based on Qt's QMainWindow framework	Divine Move (ours)	0:3	41	Divine Move (ours)
2	Divine Move (ours)	Amazons based on Qt's QMainWindow framework	3:0	39	Divine Move (ours)
3	Gomoku	Divine Move (ours)	0:3	39	Divine Move (ours)
4	Divine Move (ours)	Gomoku	3:0	36	Divine Move (ours)
5	Othello	Divine Move (ours)	0:3	35	Divine Move (ours)
6	Divine Move (ours)	Othello	3:0	33	Divine Move (ours)
7	God's Algorithm	Divine Move (ours)	0:3	43	Divine Move (ours)

Table A2. Cont.

ID	First Player	Second Player	Result	Match Point Rounds	Winner
8	Divine Move (ours)	God's Algorithm	3:0	42	Divine Move (ours)
9	Amazons Gaming System	Divine Move (ours)	0:3	39	Divine Move (ours)
10	Divine Move (ours)	Amazons Gaming System	3:0	37	Divine Move (ours)
11	Advance Little Queens	Divine Move (ours)	0:3	40	Divine Move (ours)
12	Divine Move (ours)	Advance Little Queens	3:0	37	Divine Move (ours)
13	Checkmate	Divine Move (ours)	0:3	41	Divine Move (ours)
14	Divine Move (ours)	Checkmate	3:0	38	Divine Move (ours)
15	Cliffhanger Amazons	Divine Move (ours)	0:3	40	Divine Move (ours)
16	Divine Move (ours)	Cliffhanger Amazons	3:0	37	Divine Move (ours)
17	Qi Kaide's Victory	Divine Move (ours)	0:3	41	Divine Move (ours)
18	Divine Move (ours)	Qi Kaide's Victory	3:0	36	Divine Move (ours)
19	Super Amazon	Divine Move (ours)	0:3	39	Divine Move (ours)
20	Divine Move (ours)	Super Amazon	3:0	36	Divine Move (ours)
21	God's Algorithm	Divine Move (ours)	0:3	41	Divine Move (ours)
22	Divine Move (ours)	God's Algorithm	3:0	37	Divine Move (ours)
23	Traveler Amazons	Divine Move (ours)	0:3	40	Divine Move (ours)
24	Divine Move (ours)	Traveler Amazons	3:0	35	Divine Move (ours)
25	Amazon Supreme Chess	Divine Move (ours)	0:3	41	Divine Move (ours)
26	Divine Move (ours)	Amazon Supreme Chess	3:0	39	Divine Move (ours)
27	Chess Troops Falling from Heaven	Divine Move (ours)	0:3	43	Divine Move (ours)
28	Divine Move (ours)	Chess Troops Falling from Heaven	3:0	41	Divine Move (ours)
29	Clove Amazons	Divine Move (ours)	0:3	40	Divine Move (ours)
30	Divine Move (ours)	Clove Amazons	3:0	37	Divine Move (ours)
31	Shao Guang's Amazons	Divine Move (ours)	0:3	43	Divine Move (ours)
32	Divine Move (ours)	Shao Guang's Amazons	3:0	39	Divine Move (ours)
33	AI plays chess	Divine Move (ours)	0:3	43	Divine Move (ours)
34	Divine Move (ours)	AI plays chess	3:0	39	Divine Move (ours)
35	Canopus One	Divine Move (ours)	0:3	39	Divine Move (ours)
36	Divine Move (ours)	Canopus One	3:0	35	Divine Move (ours)
37	Win the Opening Team	Divine Move (ours)	0:3	41	Divine Move (ours)
38	Divine Move (ours)	Win the Opening Team	3:0	39	Divine Move (ours)
39	Wukong Amazons	Divine Move (ours)	0:3	34	Divine Move (ours)
40	Divine Move (ours)	Wukong Amazons	3:0	30	Divine Move (ours)
41	Traveler Amazons	Divine Move (ours)	0:3	44	Divine Move (ours)
42	Divine Move (ours)	Traveler Amazons	3:0	40	Divine Move (ours)
43	Yue	Divine Move (ours)	0:3	41	Divine Move (ours)
44	Divine Move (ours)	Yue	3:0	40	Divine Move (ours)
45	Final Amazon	Divine Move (ours)	0:3	41	Divine Move (ours)
46	Divine Move (ours)	Final Amazon	3:0	38	Divine Move (ours)
47	Information University - Monte Carlo	Divine Move (ours)	1:2	42	Divine Move (ours)
48	Divine Move (ours)	Information University - Monte Carlo	3:0	40	Divine Move (ours)
49	Haha Hi	Divine Move (ours)	1:2	42	Divine Move (ours)
50	Divine Move (ours)	Haha Hi	3:0	39	Divine Move (ours)
51	Dragon Victory	Divine Move (ours)	0:3	41	Divine Move (ours)
52	Divine Move (ours)	Dragon Victory	3:0	37	Divine Move (ours)
53	Super Amazon	Divine Move (ours)	0:3	43	Divine Move (ours)
54	Divine Move (ours)	Super Amazon	3:0	38	Divine Move (ours)
55	Base Pairing Team	Divine Move (ours)	0:3	44	Divine Move (ours)
56	Divine Move (ours)	Base Pairing Team	3:0	41	Divine Move (ours)
57	Pass the Level	Divine Move (ours)	3:0	47	Pass the Level

Table A2. Cont.

ID	First Player	Second Player	Result	Match Point Rounds	Winner
58	Divine Move (ours)	Pass the Level	0:3	47	Pass the Level
59	Dalian Jiaotong University Amazons Team 1	Divine Move (ours)	0:3	41	Divine Move (ours)
60	Divine Move (ours)	Dalian Jiaotong University Amazons Team 1	3:0	39	Divine Move (ours)
61	Get Ashore	Divine Move (ours)	0:3	41	Divine Move (ours)
62	Divine Move (ours)	Get Ashore	3:0	40	Divine Move (ours)
63	Empty	Divine Move (ours)	0:3	44	Divine Move (ours)
64	Divine Move (ours)	Empty	3:0	41	Divine Move (ours)
65	Bull	Divine Move (ours)	0:3	42	Divine Move (ours)
66	Divine Move (ours)	Bull	3:0	40	Divine Move (ours)
67	Wukong Amazons	Divine Move (ours)	1:2	34	Divine Move (ours)
68	Divine Move (ours)	Wukong Amazons	3:0	31	Divine Move (ours)
69	Gangzi Fans Support Team	Divine Move (ours)	0:3	41	Divine Move (ours)
70	Divine Move (ours)	Gangzi Fans Support Team	3:0	39	Divine Move (ours)
71	Thai Pants Spicy	Divine Move (ours)	0:3	44	Divine Move (ours)
72	Divine Move (ours)	Thai Pants Spicy	3:0	41	Divine Move (ours)
73	Green Grass Cake	Divine Move (ours)	0:3	34	Divine Move (ours)
74	Divine Move (ours)	Green Grass Cake	3:0	30	Divine Move (ours)
75	Failed to Grab the Air and Didn't Grab the Plant Brain Hypoxia Team	Divine Move (ours)	0:3	40	Divine Move (ours)
76	Divine Move (ours)	Failed to Grab the Air and Didn't Grab the Plant Brain Hypoxia Team	3:0	37	Divine Move (ours)
77	DG	Divine Move (ours)	0:3	44	Divine Move (ours)
78	Divine Move (ours)	DG	3:0	39	Divine Move (ours)
79	Why is Tang Yang a God	Divine Move (ours)	0:3	42	Divine Move (ours)
80	Divine Move (ours)	Why is Tang Yang a God	3:0	39	Divine Move (ours)
81	Dream Team	Divine Move (ours)	0:3	43	Divine Move (ours)
82	Divine Move (ours)	Dream Team	3:0	39	Divine Move (ours)
83	Horse Face Skirt Daily and Easy to Wear	Divine Move (ours)	0:3	43	Divine Move (ours)
84	Divine Move (ours)	Horse Face Skirt Daily and Easy to Wear	3:0	40	Divine Move (ours)
85	Genshin Impact Expert	Divine Move (ours)	0:3	41	Divine Move (ours)
86	Divine Move (ours)	Genshin Impact Expert	3:0	37	Divine Move (ours)
87	Code Apprentice	Divine Move (ours)	0:3	42	Divine Move (ours)
88	Divine Move (ours)	Code Apprentice	3:0	41	Divine Move (ours)
89	Amazon Drift Notes	Divine Move (ours)	0:3	39	Divine Move (ours)
90	Divine Move (ours)	Amazon Drift Notes	3:0	37	Divine Move (ours)
91	Love Will Disappear, Right?	Divine Move (ours)	0:3	38	Divine Move (ours)
92	Divine Move (ours)	Love Will Disappear, Right?	3:0	33	Divine Move (ours)
93	Little Su and the Cat	Divine Move (ours)	0:3	43	Divine Move (ours)
94	Divine Move (ours)	Little Su and the Cat	3:0	42	Divine Move (ours)
95	Don't Want Marla	Divine Move (ours)	0:3	44	Divine Move (ours)
96	Divine Move (ours)	Don't Want Marla	3:0	43	Divine Move (ours)
97	Parameter Adjustment Team	Divine Move (ours)	0:3	44	Divine Move (ours)
98	Divine Move (ours)	Parameter Adjustment Team	3:0	41	Divine Move (ours)
99	AlphaAmazon	Divine Move (ours)	0:3	43	Divine Move (ours)
100	Divine Move (ours)	AlphaAmazon	3:0	42	Divine Move (ours)

Table A3. Complete game information (MCTS).

ID	First Player	Second Player	Result	Match Point Rounds	Winner
1	Amazons based on Qt's QMainWindow framework	MCTS (baseline)	1:2	43	MCTS (baseline)
2	MCTS (baseline)	Amazons based on Qt's QMainWindow framework	3:0	41	MCTS (baseline)
3	Gomoku	MCTS (baseline)	3:0	47	Gomoku
4	MCTS (baseline)	Gomoku	3:0	45	MCTS (baseline)
5	Othello	MCTS (baseline)	2:1	47	Othello
6	MCTS (baseline)	Othello	2:1	44	MCTS (baseline)
7	God's Algorithm	MCTS (baseline)	0:3	43	MCTS (baseline)
8	MCTS (baseline)	God's Algorithm	3:0	40	MCTS (baseline)
9	Amazons Gaming System	MCTS (baseline)	0:3	42	MCTS (baseline)
10	MCTS (baseline)	Amazons Gaming System	3:0	40	MCTS (baseline)
11	Advance Little Queens	MCTS (baseline)	3:0	47	Advance Little Queens
12	MCTS (baseline)	Advance Little Queens	3:0	43	MCTS (baseline)
13	Checkmate	MCTS (baseline)	2:1	47	Checkmate
14	MCTS (baseline)	Checkmate	3:0	40	MCTS (baseline)
15	Cliffhanger Amazons	MCTS (baseline)	3:0	47	Checkmate
16	MCTS (baseline)	Cliffhanger Amazons	3:0	44	MCTS (baseline)
17	Qi Kaide's Victory	MCTS (baseline)	0:3	44	MCTS (baseline)
18	MCTS (baseline)	Qi Kaide's Victory	3:0	41	MCTS (baseline)
19	Super Amazon	MCTS (baseline)	2:1	47	Super Amazon
20	MCTS (baseline)	Super Amazon	2:1	43	MCTS (baseline)
21	God's Algorithm	MCTS (baseline)	3:0	47	God's Algorithm
22	MCTS (baseline)	God's Algorithm	2:1	40	MCTS (baseline)
23	Traveler Amazons	MCTS (baseline)	2:1	47	Traveler Amazons
24	MCTS (baseline)	Traveler Amazons	3:0	42	MCTS (baseline)
25	Amazon Supreme Chess	MCTS (baseline)	0:3	40	MCTS (baseline)
26	MCTS (baseline)	Amazon Supreme Chess	3:0	41	MCTS (baseline)
27	Chess Troops Falling from Heaven	MCTS (baseline)	0:3	43	MCTS (baseline)
28	MCTS (baseline)	Chess Troops Falling from Heaven	3:0	39	MCTS (baseline)
29	Clove Amazons	MCTS (baseline)	0:3	44	MCTS (baseline)
30	MCTS (baseline)	Clove Amazons	3:0	40	MCTS (baseline)
31	Shao Guang's Amazons	MCTS (baseline)	1:2	43	MCTS (baseline)
32	MCTS (baseline)	Shao Guang's Amazons	3:0	40	MCTS (baseline)
33	AI plays chess	MCTS (baseline)	2:1	47	AI plays chess
34	MCTS (baseline)	AI plays chess	3:0	42	MCTS (baseline)
35	Canopus One	MCTS (baseline)	0:3	43	MCTS (baseline)
36	MCTS (baseline)	Canopus One	2:1	40	MCTS (baseline)
37	Win the Opening Team	MCTS (baseline)	3:0	47	Win the Opening Team
38	MCTS (baseline)	Win the Opening Team	3:0	44	MCTS (baseline)
39	Wukong Amazons	MCTS (baseline)	0:3	42	MCTS (baseline)
40	MCTS (baseline)	Wukong Amazons	3:0	39	MCTS (baseline)
41	Traveler Amazons	MCTS (baseline)	0:3	41	MCTS (baseline)
42	MCTS (baseline)	Traveler Amazons	3:0	36	MCTS (baseline)
43	Yue	MCTS (baseline)	0:3	44	MCTS (baseline)
44	MCTS (baseline)	Yue	3:0	41	MCTS (baseline)
45	Final Amazon	MCTS (baseline)	0:3	42	MCTS (baseline)
46	MCTS (baseline)	Final Amazon	3:0	37	MCTS (baseline)
47	Information University - Monte Carlo	MCTS (baseline)	3:0	47	Information University-Monte Carlo
48	MCTS (baseline)	Information University-Monte Carlo	0:3	47	Information University-Monte Carlo

Table A3. Cont.

ID	First Player	Second Player	Result	Match Point Rounds	Winner
49	Haha Hi	MCTS (baseline)	3:0	47	Haha Hi
50	MCTS (baseline)	Haha Hi	3:0	41	MCTS (baseline)
51	Dragon Victory	MCTS (baseline)	0:3	43	MCTS (baseline)
52	MCTS (baseline)	Dragon Victory	3:0	38	MCTS (baseline)
53	Super Amazon	MCTS (baseline)	0:3	41	MCTS (baseline)
54	MCTS (baseline)	Super Amazon	3:0	38	MCTS (baseline)
55	Base Pairing Team	MCTS (baseline)	3:0	45	Base Pairing Team
56	MCTS (baseline)	Base Pairing Team	0:3	42	Base Pairing Team
57	Pass the Level	MCTS (baseline)	3:0	47	Pass the Level
58	MCTS (baseline)	Pass the Level	0:3	47	Pass the Level
59	Dalian Jiaotong University Amazons Team 1	MCTS (baseline)	0:3	44	MCTS (baseline)
60	MCTS (baseline)	Dalian Jiaotong University Amazons Team 1	3:0	42	MCTS (baseline)
61	Get Ashore	MCTS (baseline)	3:0	47	Get Ashore
62	MCTS (baseline)	Get Ashore	0:3	47	Get Ashore
63	Empty	MCTS (baseline)	0:3	45	MCTS (baseline)
64	MCTS (baseline)	Empty	3:0	41	MCTS (baseline)
65	Bull	MCTS (baseline)	0:3	45	MCTS (baseline)
66	MCTS (baseline)	Bull	3:0	40	MCTS (baseline)
67	Wukong Amazons	MCTS (baseline)	2:1	47	Wukong Amazons
68	MCTS (baseline)	Wukong Amazons	3:0	39	MCTS (baseline)
69	Gangzi Fans Support Team	MCTS (baseline)	3:0	44	Gangzi Fans Support Team
70	MCTS (baseline)	Gangzi Fans Support Team	1:2	41	Gangzi Fans Support Team
71	Thai Pants Spicy	MCTS (baseline)	0:3	42	MCTS (baseline)
72	MCTS (baseline)	Thai Pants Spicy	3:0	40	MCTS (baseline)
73	Green Grass Cake	MCTS (baseline)	0:3	42	MCTS (baseline)
74	MCTS (baseline)	Green Grass Cake	3:0	37	MCTS (baseline)
75	Failed to Grab the Air and Didn't Grab the Plant Brain Hypoxia Team	MCTS (baseline)	0:3	41	MCTS (baseline)
76	MCTS (baseline)	Failed to Grab the Air and Didn't Grab the Plant Brain Hypoxia Team	3:0	40	MCTS (baseline)
77	DG	MCTS (baseline)	0:3	43	MCTS (baseline)
78	MCTS (baseline)	DG	3:0	39	MCTS (baseline)
79	Why is Tang Yang a God	MCTS (baseline)	2:1	47	Why is Tang Yang a God
80	MCTS (baseline)	Why is Tang Yang a God	3:0	44	MCTS (baseline)
81	Dream Team	MCTS (baseline)	0:3	43	MCTS (baseline)
82	MCTS (baseline)	Dream Team	3:0	44	MCTS (baseline)
83	Horse Face Skirt Daily and Easy to Wear	MCTS (baseline)	0:3	44	MCTS (baseline)
84	MCTS (baseline)	Horse Face Skirt Daily and Easy to Wear	3:0	40	MCTS (baseline)
85	Genshin Impact Expert	MCTS (baseline)	0:3	44	MCTS (baseline)
86	MCTS (baseline)	Genshin Impact Expert	3:0	39	MCTS (baseline)
87	Code Apprentice	MCTS (baseline)	0:3	42	MCTS (baseline)
88	MCTS (baseline)	Code Apprentice	3:0	40	MCTS (baseline)
89	Amazon Drift Notes	MCTS (baseline)	0:3	44	MCTS (baseline)
90	MCTS (baseline)	Amazon Drift Notes	3:0	42	MCTS (baseline)
91	Love Will Disappear, Right?	MCTS (baseline)	0:3	40	MCTS (baseline)
92	MCTS (baseline)	Love Will Disappear, Right?	3:0	38	MCTS (baseline)

Table A3. Cont.

ID	First Player	Second Player	Result	Match Point Rounds	Winner
93	Little Su and the Cat	MCTS (baseline)	0:3	44	MCTS (baseline)
94	MCTS (baseline)	Little Su and the Cat	3:0	40	MCTS (baseline)
95	Don't Want Marla	MCTS (baseline)	0:3	41	MCTS (baseline)
96	MCTS (baseline)	Don't Want Marla	3:0	37	MCTS (baseline)
97	Parameter Adjustment Team	MCTS (baseline)	0:3	42	MCTS (baseline)
98	MCTS (baseline)	Parameter Adjustment Team	3:0	39	MCTS (baseline)
99	AlphaAmazon	MCTS (baseline)	0:3	41	MCTS (baseline)
100	MCTS (baseline)	AlphaAmazon	0:3	47	AlphaAmazon

References

- Li, R.; Gao, M. Amazons search algorithm design based on CNN model. *Digit. Technol. Appl.* **2022**, *40*, 164–166.
- Guo, Q.; Li, S.; Bao, H. Research on evaluation function computer game of Amazon. *Comput. Eng. Appl.* **2012**, *48*, 50–54.
- Guo, T.; Qiu, H.; Tong, B.; Wang, Y. Optimization and Comparison of Multiple Game Algorithms in Amazons. In Proceedings of the 2019 Chinese Control And Decision Conference (CCDC), Nanchang, China, 3–5 June 2019; pp. 6299–6304.
- Quan, J.; Qiu, H.; Wang, Y.; Li, F.; Qiu, S. Application of UCT technologies for computer games of Amazon. In Proceedings of the 2016 Chinese Control and Decision Conference (CCDC), Yinchuan, China, 28–30 May 2016; pp. 6896–6899.
- Ju, J.; Qiu, H.; Wang, F.; Wang, X.; Wang, Y. Research on Thread Optimization and Opening Library Based on Parallel PVS Algorithm in Amazons. In Proceedings of the 2021 33rd Chinese Control and Decision Conference (CCDC), Kunming, China, 28–30 May 2021; pp. 2207–2212.
- Li, Z.; Ning, C.; Cao, J.; Li, Z. Amazons Based on UCT-PVS Hybrid Algorithm. In Proceedings of the 2021 33rd Chinese Control and Decision Conference (CCDC), Kunming, China, 28–30 May 2021; pp. 2179–2183.
- Ding, M.; Bo, J.; Qi, Y.; Fu, Y.; Li, S. Design of Amazons Game System Based on Reinforcement Learning. In Proceedings of the 2019 Chinese Control And Decision Conference (CCDC), Nanchang, China, 3–5 June 2019; pp. 6337–6342.
- Tong, B.; Qiu, H.; Guo, T.; Wang, Y. Research and Application of Parallel Computing of PVS Algorithm Based on Amazon Human-Machine Game. In Proceedings of the 2019 Chinese Control And Decision Conference (CCDC), Nanchang, China, 3–5 June 2019; pp. 6293–6298.
- Chen, X.; Yang, L. Research on evaluation function in Amazons. *Comput. Knowl. Technol.* **2019**, *15*, 224–226.
- Wang, C.; Ding, M. Interface design and implementation of personalized Amazons. *Intell. Comput. Appl.* **2017**, *7*, 78–80.
- Zhang, L. Research on Amazons Game System Based on Minimax Search Algorithm. Master's Thesis, Northeast University, Shenyang, China, 2010.
- Metropolis, N.; Ulam, S. The Monte Carlo Method. *J. Am. Stat. Assoc.* **1949**, *44*, 335–341. [[CrossRef](#)] [[PubMed](#)]
- Coulom, R. Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search. In Proceedings of the Computers and Games, Turin, Italy, 1–5 April 2007; pp. 72–83.
- Gelly, S.; Kocsis, L.; Schoenauer, M.; Sebag, M.; Silver, D.; Szepesvári, C.; Teytaud, O. The grand challenge of computer Go: Monte Carlo tree search and extensions. *Commun. ACM* **2012**, *55*, 106–113. [[CrossRef](#)]
- Gelly, S.; Silver, D. Monte-Carlo tree search and rapid action value estimation in computer Go. *Artif. Intell.* **2011**, *175*, 1856–1875. [[CrossRef](#)]
- Vinyals, O.; Babuschkin, I.; Czarnecki, W.M.; Mathieu, M.; Dudzik, A.; Chung, J.; Choi, D.H.; Powell, R.; Ewalds, T.; Georgiev, P.; et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* **2019**, *575*, 350–354. [[CrossRef](#)]
- Silver, D.; Huang, A.; Maddison, C.J.; Guez, A.; Sifre, L.; van den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. Mastering the game of Go with deep neural networks and tree search. *Nature* **2016**, *529*, 484–489. [[CrossRef](#)]
- Berner, C.; Brockman, G.; Chan, B.; Cheung, V.; Debiak, P.; Dennison, C.; Farhi, D.; Fischer, Q.; Hashme, S.; Hesse, C.; et al. Dota 2 with Large Scale Deep Reinforcement Learning. *arXiv* **2019**, arXiv:1912.06680.
- Browne, C.B.; Powley, E.; Whitehouse, D.; Lucas, S.M.; Cowling, P.I.; Rohlfshagen, P.; Tavener, S.; Perez, D.; Samothrakis, S.; Colton, S. A Survey of Monte Carlo Tree Search Methods. *IEEE Trans. Comput. Intell. AI Games* **2012**, *4*, 1–43. [[CrossRef](#)]
- Kloetzer, J.; Iida, H.; Bouzy, B. The Monte-Carlo Approach in Amazons. In Proceedings of the Computer Games Workshop, Amsterdam, The Netherlands, 15–17 June 2007; pp. 185–192.
- Shannon, C. E. Programming a computer for playing chess. *Lond. Edinb. Dublin Philos. Mag. J. Sci.* **1950**, *41*, 256–275. [[CrossRef](#)]
- Chaslot, G.; Bakkes, S.; Szita, I.; Spronck, P. Monte-Carlo Tree Search: A New Framework for Game AI. In Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, California, CA, USA, 22–24 October 2008; pp. 216–217.
- Świechowski, M.; Godlewski, K.; Sawicki, B.; Mańdziuk, J. Monte Carlo Tree Search: A review of recent modifications and applications. *Artif. Intell. Rev.* **2023**, *56*, 2497–2562. [[CrossRef](#)]

24. Kloetzer, J. Monte-Carlo Opening Books for Amazons. In *Computers and Games: 7th International Conference (CG 2010), Kanazawa, Japan, September 24–26 2010, Revised Selected Papers 7*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 124–135.
25. Song, J.; Müller, M. An Enhanced Solver for the Game of Amazons. *IEEE Trans. Comput. Intell. AI Games* **2014**, *7*, 16–27. [[CrossRef](#)]
26. Zhang, G.; Chen, X.; Chang, R.; Zhang, Y.; Wang, C.; Bai, L.; Wang, J.; Xu, C. Mastering the Game of Amazons Fast by Decoupling Network Learning. In Proceedings of the 2021 International Joint Conference on Neural Networks (IJCNN), Shenzhen, China, 18–22 July 2021; pp. 1–8.
27. de Koning, J. INVADER Prolongs Amazons Title. *ICGA J.* **2011**, *34*, 96. [[CrossRef](#)]
28. Kloetzer, J. INVADER Wins Amazons Tournament. *ICGA J.* **2009**, *32*, 112–113. [[CrossRef](#)]
29. Lorentz, R. Invader Wins Eighth Amazons Gold Medal. *ICGA J.* **2017**, *39*, 228–229. [[CrossRef](#)]
30. Guo, Q.; Li, S.; Bao, H. The Research of Searching Algorithm in Amazons Game. In Proceedings of the 2011 Chinese Control and Decision Conference (CCDC), Mianyang, China, 23–25 May 2011; pp. 1859–1862.
31. Li, X.; Hou, L.; Wu, L. UCT Algorithm in Amazons Human-Computer Games. In Proceedings of the 26th Chinese Control and Decision Conference (CCDC), Changsha, China, 31 May–2 June 2014; pp. 3358–3361.
32. Quan, J.; Qiu, H.; Wang, Y.; Li, Y.; Wang, X. Study the Performance of Search Algorithms in Amazons. In Proceedings of the 27th Chinese Control and Decision Conference (CCDC), Qingdao, China, 23–25 May 2015; pp. 5811–5813.
33. Lieberum, J. An Evaluation Function for the Game of Amazons. *Theor. Comput. Sci.* **2005**, *349*, 230–244. [[CrossRef](#)]
34. Chai, Z.; Fang, Z.; Zhu, J. Amazons Evaluation Optimization Strategy Based on PSO Algorithm. In Proceedings of the 2019 Chinese Control And Decision Conference (CCDC), Nanchang, China, 3–5 June 2019; pp. 6334–6336.
35. Sun, Y.; Yuan, D.; Gao, M.; Zhu, P. GPU Acceleration of Monte Carlo Tree Search Algorithm for Amazons and Its Evaluation Function. In Proceedings of the 2022 International Conference on Artificial Intelligence, Information Processing and Cloud Computing (AIIPCC), Kunming, China, 21–23 June 2022; pp. 434–440.
36. Kocsis, L.; Szepesvari, C.; Willemson, J. Improved Monte-Carlo Search. *Univ. Tartu Est. Tech. Rep.* **2006**, *1*, 1–22.
37. Finnsson, H.; Björnsson, Y. Game-tree properties and MCTS performance. *IJCAI* **2011**, *11*, 23–30.
38. Pheatt, C. Intel® threading building blocks. *J. Comput. Sci. Coll.* **2008**, *23*, 164–166.
39. Childs, B.E.; Brodeur, J.H.; Kocsis, L. Transpositions and move groups in monte carlo tree search. In Proceedings of the 2008 IEEE Symposium on Computational Intelligence and Games, Perth, Australia, 15–18 December 2008; pp. 389–395.
40. Van Eyck, G.; Müller, M. Revisiting move groups in monte-carlo tree search. In *Advances in Computer Games: 13th International Conference, ACG 2011, Tilburg, The Netherlands, 20–22 November 2011, Revised Selected Papers 13*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 13–23.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.

Article

Color Standardization of Chemical Solution Images Using Template-Based Histogram Matching in Deep Learning Regression

Patrycja Kwiek  and Małgorzata Jakubowska * 

Faculty of Materials Science and Ceramics, AGH University of Krakow, al. Mickiewicza 30, 30-059 Kraków, Poland; pakwiek@agh.edu.pl

* Correspondence: jakubows@agh.edu.pl

Abstract: Color distortion in an image presents a challenge for machine learning classification and regression when the input data consists of pictures. As a result, a new algorithm for color standardization of photos is proposed, forming the foundation for a deep neural network regression model. This approach utilizes a self-designed color template that was developed based on an initial series of studies and digital imaging. Using the equalized histogram of the R, G, B channels of the digital template and its photo, a color mapping strategy was computed. By applying this approach, the histograms were adjusted and the colors of photos taken with a smartphone were standardized. The proposed algorithm was developed for a series of images where the entire surface roughly maintained a uniform color and the differences in color between the photographs of individual objects were minor. This optimized approach was validated in the colorimetric determination procedure of vitamin C. The dataset for the deep neural network in the regression variant was formed from photos of samples under two separate lighting conditions. For the vitamin C concentration range from 0 to $87.72 \mu\text{g}\cdot\text{mL}^{-1}$, the RMSE for the test set ranged between 0.75 and $1.95 \mu\text{g}\cdot\text{mL}^{-1}$, in comparison to the non-standardized variant, where this indicator was at the level of $1.48\text{--}2.29 \mu\text{g}\cdot\text{mL}^{-1}$. The consistency of the predicted concentration results with actual data, expressed as R^2 , ranged between 0.9956 and 0.9999 for each of the standardized variants. This approach allows for the removal of light reflections on the shiny surfaces of solutions, which is a common problem in liquid samples. This color-matching algorithm has universal character, and its scope of application is not limited.



Citation: Kwiek, P.; Jakubowska, M. Color Standardization of Chemical Solution Images Using Template-Based Histogram Matching in Deep Learning Regression. *Algorithms* **2024**, *17*, 335. <https://doi.org/10.3390/a17080335>

Academic Editor: Marcin Iwanowski

Received: 11 July 2024

Revised: 29 July 2024

Accepted: 30 July 2024

Published: 1 August 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

Keywords: color standardization; color template matching; RGB histogram fitting; deep learning; vitamin C colorimetric determination

1. Introduction

The development of measurement methods and research strategies across various fields of science and technology is closely linked to the potential support offered by machine learning (ML) and deep learning (DL) techniques. If the information in the problem being solved is communicated through images, drawings, sketches, or photographs, the use of DL in data modeling is often the optimal approach. The application of neural networks in typical supervised learning requires access to large data sets, which makes the method of obtaining and labeling data a significant limitation. Today's technology provides many electronic devices, such as cameras or scanners, which can be used to obtain data for constructing models quickly and without special qualifications. Information can often be encoded simply in the form of objects, the relationship between objects in the analyzed area, textures, the quantity of small components, and colors.

Several color spaces are widely used across different industries for various applications [1]. RGB space, based on primary colors of light (red, green, and blue), is extensively used in digital displays and imaging. CMYK (cyan, magenta, yellow, and black) is used in color printing, relying on the subtractive color model to produce a wide range of colors

on paper. HSI/HSV (hue, saturation, intensity/value) is often used in image analysis and computer vision due to its alignment with human perception of colors. YUV and YCbCr are used in video compression and broadcasting, separating the image into luminance and chrominance components to optimize data storage and transmission. The CIE Lab color space, developed by the International Commission on Illumination, provides a perceptually uniform color representation, making it ideal for color differentiation and transformation tasks. RAL CLASSIC is a widely used color matching system in Europe, consisting of 213 standardized colors, while RAL DESIGN offers a more extensive palette of 1625 colors for designers and architects. RAL DIGITAL facilitates precise color communication in digital applications through RGB and CMYK values. The PANTONE Matching System (PMS) is globally recognized in graphic design, printing, and fashion, providing standardized color codes to ensure consistency across different media and materials. Each of these color spaces has its own advantages and applications, catering to the specific needs of various industries.

If color is the main, decisive criterion in image analysis that codes the information we are looking for, it is crucial to take photos in such a way as to maintain the appropriate quality of this parameter. The true color of the image allows for the correct interpretation of various features of the photographed objects. Undesirable variability may arise from the use of multiple devices to capture images or from the method of acquiring images with a single device under uncontrolled or changing conditions. Significant factors include changes in lighting conditions, reflections on shiny surfaces, and variations such as the time of day. These elements may not directly impact the photo's quality but can influence its informational value. Therefore, the importance of color standardization as a crucial step in data preprocessing should be considered. Various computational approaches for preserving or restoring the true color of images within a dataset for deep learning are described in numerous works, offering solutions to this problem.

This study presents an original algorithm for standardizing the colors of photos captured with a single smartphone camera under various lighting conditions. It also considers variability resulting from incidental factors associated with the photography process and the inherent lack of perfect repeatability of the camera. The research was focused on a homogeneous liquid that showed minimal color variation among the tested objects. The entirety of the photo used in the modeling process shared the same color, although with a certain level of random variability that could not be entirely eliminated. In this context, even slight changes in color held significant information value, emphasizing the importance of accurate color interpretation for enhancing the quality of the deep learning regression model and minimizing prediction errors.

A standardization algorithm was proposed that used the histogram of a specially designed template photographed under controlled conditions together with the object tested in relation to the histogram of the template prepared using a graphic application and its photograph. This study also presents the verification of the proposed approach for the colorimetric determination of vitamin C using a regression model defined by DNN (deep neural networks).

2. Related Works

2.1. Color Standardization

There are numerous categories of color calibration problems, with two fundamental groups pertaining to either the use of different devices within a series of photos or variations in the lighting conditions affecting the photographed subjects. Hardware color compensation can be achieved by employing camera sensors that automatically adjust settings based on the lighting of the scene being captured. The alternative approach involves numerical processing of photos using various strategies that ultimately aim to achieve consistent color tones across multiple images. To accomplish this, calibration charts are initially employed, which correlate and map the individual pixels of captured photos with the standardized values of the corresponding color patches on the chart [2,3]. A standard color

calibration chart serves as a reference tool in photography, film creation, video recording, or graphic design, ensuring accurate and consistent color reproduction in images. Typically, it comprises a grid of colored patches with known and standardized colors. These charts are applied in calibrating equipment and adjusting during post-processing, thereby ensuring consistency in color representation across various devices and conditions. Commercially, several calibration charts are available, ranging from those with six to 100 color patches. However, the 24-color patch calibration chart, often referred to by its original name as the Macbeth ColorChecker, has found widespread use in numerous applications [4]. Additionally, the development of customized calibration charts plays an important role in standardizing image colors.

Color standardization plays an important role in many fields of science and technology, including medical diagnostics, bioanalysis, agriculture, analysis of evidence in forensic sciences, analytical chemistry, etc. In agricultural image processing, the focus lies in linking alterations in the color of images, in designated areas of interest, with particular quality attributes such as plant health, crop stress, and maturity level. In [5], a technique was proposed to calibrate digital images, ensuring their uniformity and improving phenological comparisons in agriculture. The color transformation of the images was achieved using a standard 24-color calibration chart. Calibration delivers typical and expected results by adjusting the RGB values of the color patches in the input image to standard RGB values. In [6], a digital shade-matching device was developed for determining tooth color, as accurate shade assessment is crucial in dental restoration procedures. Image recordings made with an intraoral camera served as the basis for the calculations and transformations. The support vector machine (SVM) algorithm and the CIEL*a*b* color space were used to evaluate tooth color. The color correction was based on the calculation of the Euclidean distance between the shades. The authors created a dedicated template, distinct from the typical one with a broad spectrum of colors, specifically adapted to the range of tooth color variations addressed in the study. The next proposition concerns improving the accuracy of color reproduction in digital cameras and smartphones. The first method uses white balance correction followed by a color space transformation. The second approach, applicable to images processed offline, is based on performing full color balance instead of white balance and requires the use of machine learning to predict the color balance matrix [7]. The review paper [8] discusses the use of various histopathological image recognition algorithms. The differences in color in histopathology result from the use of various equipment or tissue staining techniques. Typically, color normalization involves transferring the average color of the target image to the source image and separating the stain present in the source image. The article describes an extensive set of color standardization methods that have been verified using histopathological images, including those obtained in the diagnosis of breast and colon cancer.

Due to the development of digital imaging technology and consequently the emergence of many completely new applications, solutions based on smartphone images are being proposed in various fields. Table 1 contains a list of selected articles in which color images serve as important information carriers, and where color correction and standardization significantly impact modeling results and the quality of the proposed solutions.

Table 1. Comparison of related works.

Ref.	Year	Problem Solved	Dataset Used	Color Correction Strategy
[9]	2022	The general concept of relative color constancy (RCC) i.e., the ability to align colors of the same objects between images independent of illumination and camera	Three image datasets collected by multiple cameras under various illumination and exposure conditions (Belfast, Middlebury, Gehler-Shi)	Three steps of color alignment framework: camera response calibration with color patches, response linearization, pixel-wise color intensity, and chromaticity matching

Table 1. Cont.

Ref.	Year	Problem Solved	Dataset Used	Color Correction Strategy
[10]	2022	Color variations within clinical images of maxillofacial prosthetic silicone specimens (in vitro approach)	Images of the pigmented prosthetic silicone specimens within different ambient lighting conditions	Computerized software-based post-processing white balance calibration (PPWBC) using a gray card and Macbeth color chart
[11]	2023	Non-invasive cultural heritage conservation in the context of material degradation	Four sets of images in different lighting conditions, acquired with a conventional cellphone camera in sRGB color space	An adaptive transfer function, working with a self-made color calibration chart; neural network-based final tuning of the calibration functions
[12]	2023	Smartphone-based urine colorimetry	Urine samples from 58 patients photographed in a customized photo box, under five simulated lighting conditions, using five smartphones in RGB color space	Color calibration with SpyderCHEKR TM color chart and the Adobe Photoshop proprietary algorithms to scale the uncorrected RGB values
[13]	2023	A smartphone-based solution for accurate colorimetric measurements, with an augmented-reality guiding system	Simulated and real images, images of color stripe kits for pH reading	A color correction algorithm with the first-order spatial varying regression model which leverages both the absolute color magnitude and scale; the system includes a color reference board
[14]	2023	Assessment of the healing process of chronic wounds	Wound tissue images taken at different times, under different lighting conditions, distances, lenses, and smartphone cameras in RGB color space	Images of wound tissue taken with a Macbeth chart of 24 square color patches and four ArUco markers at corners of the chart; transformation matrix calculation using the Moore–Penrose inverse matrix and the target matrix
[15]	2024	A general purpose algorithm	Photos of nature and city landscapes, vegetation, and human-made objects in RGB color space	A matrix-based transformation using tensor product B-splines with optimization of the smoothing parameter λ and number of spline basis functions
[16]	2024	Reproduction of reference color of textile clothing by selecting appropriate disperse dyes	Results of colorimetric measurements using a spectrophotometer type Spectraflash 600 Plus (Datacolor International, USA)	A genetic algorithm to optimize the selection and concentration of dyes for exact color matching in bicomponent (PET/PTT) filaments
[17]	2024	Optimization of the appearances of plant-based foods considering that animal-based products may contain a wide range of different pigments and structural elements that scatter light	L*, a*, b* values of animal-based products and plant-based emulsions produced measured by colorimeter (ColorFlex EZ 45/0-LAV, Hunter Associates Laboratory Inc., Reston, VA, USA)	The color matching model based on Kubelka–Munk theory, used to calculate the spectral reflectance of the mixed emulsions from the spectral reflectance of the individual color-loaded emulsions
[18]	2024	Color design for fashion—color selection and matching different color combinations	Graphic objects designed in CorelDraw	Color matching based on an interactive genetic algorithm using reference images

2.2. Vitamin C Determination

Color standardization techniques are crucial in analytical chemistry, especially in the colorimetric determination of different substances. This is particularly evident in methods where solutions of tested compounds are photographed, and concentration information is derived from these photos using various statistical tools and multivariate analysis. An illustration of such a methodology is the colorimetric determination of vitamin C.

Ascorbic acid (AA), i.e., vitamin C, is a natural, highly water-soluble compound, which plays a dominant role in human health and is essential for a variety of metabolic reactions and biological processes, with strong antioxidant activity, limiting free radical damage. Insufficient concentration of vitamin C in the body, among others, causes scurvy and anemia and is associated with some psychological abnormalities, for example, depression. Overconsumption of vitamin C can have negative consequences, such as impeding vitamin B12 absorption and potentially causing gastrointestinal problems or kidney stones [19,20]. As humans are unable to produce vitamin C internally, it is essential to supplement with a regulated amount of AA, typically from fresh fruits and vegetables [21,22].

Analytical chemistry provides many strategies to determine AA in food products and vitamin C supplements. Traditionally, instrumental analysis techniques have been utilized to measure ascorbic acid levels, offering a wide concentration range with a low detection limit. These methods include direct titration [23], ultraperformance liquid chromatography (UPLC), high-performance liquid chromatography (HPLC) [24,25], Fourier transform infrared spectrometry (FT-IR) [26], chemiluminescence [27], Raman spectroscopy [28], near-infrared (NIR), FT-Raman spectroscopy [29], and capillary electrophoresis [30]. Furthermore, a variety of electrochemical methods have been used to determine AA, offering extremely high sensitivity, selectivity, reproducibility, and reliability [31–33]. However, the mentioned methods of chemical instrumental analysis require specialized laboratory rooms, expensive instrumentation, many auxiliary reagents, time-consuming extraction steps, and complicated measurement procedures that require highly skilled staff for tasks such as preparing reagents and samples, conducting research, or interpreting results.

The development of novel methodologies that either substitute or support traditional laboratory strategies with cost-effective instruments fulfills the need for highly efficient and adaptable approaches in food, pharmaceutical, and environmental analysis. Popular and widely available smartphones offer a promising analytical platform characterized by full automation, large local memory capacity, access to external cloud storage, and especially high-quality cameras, making them suitable for wide use as portable measuring devices. Compared to conventional methods, smartphone-based measurements use digital imaging, offering many applications in various fields and significant advantages. These include low cost, simple operation, the possibility of installing various dedicated sensors, precise measurements in real time, a non-destructive nature of testing in many cases, fast analysis, and portability to other similar devices. In analytical chemistry, commonly available devices, such as cameras, webcams, scanners, and smartphones, are applied to register images from colorimetric reactions in which the intensity of the resulting color is directly correlated to the concentration of the analyte. The literature provides various descriptions of the use of digital imaging methods for determining AA. The work [34] presents a straightforward, rapid, and cost-effective technique employing smartphone image analysis to quantify AA in an aqueous solution. The study was based on the evaluation of the disturbing effect of AA in the enzymatic colorimetric detection of glucose. Another approach has been suggested, involving digital image colorimetry to determine the vitamin C content in natural fruit juices. This method involves measuring the color of the iron(II)—o-phenanthroline complex [35]. In the next article, the authors developed a new circular dichroism spectrometer based on smartphones for highly sensitive and ultraportable colorimetric analysis, offering the advantages of cost-effectiveness and simplicity. This spectrometer utilized an HSV color model and determined absorbance by summing the V value of adjacent positions to calculate the overall intensity. The TMB-MnO₂ nanosheet reaction was also used and, in effect, a highly sensitive and specific system was proposed for the detection of vitamin

C [36]. In [37] a point-of-care sensor was developed, consisting of a fluorescent paper chip, 3D printed parts, and a smartphone. This system enabled quantitative and highly sensitive real-time visual detection of AA. The design of this device involved placing a fluorescent sensor on a portable sensing platform connected to a smartphone.

The analytical measurement method used in this work was based on the reducing properties of vitamin C. AA reduces Fe(III) to Fe(II) and then forms a complex with o-phenanthroline. The strong red complex of Fe(II) and o-phenanthroline is widely used for colorimetric determination and as an indicator [38]. This reaction is characterized by high speed, stability, and reproducibility. Utilizing this camera-based approach coupled with digital imaging of colored compounds enhances accuracy and reduces the cost of measuring vitamin C content.

2.3. Multivariate Regression Algorithms

Regression techniques are extensively used as a supervised learning method, involving modeling the relationship between features and the target variable to address tasks that aim to predict continuous values. Typical algorithms used for this purpose are PCR (principal component regression) [39], PLSR (partial least squares regression) [40,41], and SVR (support vector regression) [42]. DNN, with various architectures that map the nonlinear structure of multivariate data, can also be applied to regression problems [43,44]. Convolutional neural networks (CNNs) are a type of deep learning algorithm whose operation is based on a mathematical operation called convolution. They consist of multiple layers of convolutional filters that automatically learn hierarchical representations of features directly from raw pixel data, making them highly effective for tasks such as image classification or segmentation and object detection. Typically, the quality of regression models is evaluated using the mean squares of the errors between real values and the predictions (MSE) or the mean absolute error (MAE).

MSE and MAE are different error metrics used to evaluate the performance of predictive models. MSE is more sensitive to large errors because they are squared, while MAE provides a more intuitive understanding of the average error and is expressed in the unit of the target variable. These scores used as loss functions during neural network training minimize the mean difference between the predicted and the actual values. The training and prediction performance of the calibration model is also evaluated based on the following scores: the root mean square error of calibration (RMSEC) and prediction (RMSEP), and the coefficient of determination R^2 for calibration and prediction [45]. Since the prediction result is expected to match the actual value, the points with coordinates (actual, predicted) should ideally lie on the line $y = x$. However, due to random variability and imperfect model fit to the data, two conditions should be met: the value of one should fall within the confidence interval of the slope, and a value of zero should fall within the confidence interval of the intercept [46].

3. Proposed Method

3.1. Color Matching Algorithm

In this work, the main objective was to develop an approach for determining vitamin C based on the color change of the solution influenced by its concentration. Therefore, a crucial step in achieving a high-quality regression model was performing the color standardization procedure on the photographed vitamin C samples. The proposed color standardization algorithm was divided into two parts. In the first step, standard histogram matching was performed between a digital color template and a photograph of a printed template obtained under a given lighting condition. In the second step, histogram matching was performed between a photo of the solution (tested sample) taken under the same illumination as the pattern and the reference template. In the end, 512 smaller random squares were cut from the reconstructed image to obtain a sufficiently large dataset for the subsequent training of the regression neural network. The number of randomly selected small squares was optimized, as preliminary calculations showed that insufficient data

resulted in lower model quality. It was also necessary to take an additional step of adapting the self-designed reference template to the appropriate value range. This operation resulted in the development of an adapted histogram-matching algorithm.

Histogram matching is an operation in which the pixels of an input image are manipulated in a way that allows its histogram to match the histogram of the reference image. In this work, the RGB color space was chosen for the calculations. It is widely available and popular, as most cameras and image capture devices record data directly in RGB format, eliminating the need for additional color space transformations. The RGB space is also simple to implement, being intuitive and directly related to the physical properties of light, which simplifies algorithm development. Additionally, in our specific application, RGB has proven to give satisfactory results. Therefore, despite the advantages of other color spaces, RGB was the most suitable for practical and technical reasons. In the case of color images, each channel (RGB) was considered independently. In the context of image processing, this was a normalization task, associated with, for example, changing illumination (Figure 1).



Figure 1. Example of histogram matching for illustrative purposes.

In mathematical terms, histogram matching can be represented as an image transformation such that the cumulative distribution function (CDF) of the values in each band corresponds to the CDF of bands in another image.

To match the histograms of two images, it is necessary to equalize their histograms. After that, each pixel of the first image is modified and mapped to the second image. This operation was performed for each of the channels (R, G, B) of a photographed template under a given lighting condition (natural light and artificial light) and for the channels of a digital template. The whole process of picture color transformation is represented in Figure 2.

In order to standardize the images of the tested chemical solutions, the correct range of RGB values was selected from the histograms of the photo of the tested sample. If the histogram value for a sample was greater than or equal to 1% of the maximum value of the chemical solution histogram, the histogram value was not modified, otherwise it was set to 0. The discrepancy in the number of pixels was neutralized by their uniform random distribution across the range of values. The final step in the standardization algorithm was to match the histograms of an image of a solution with the adapted template histograms. This was done in the same way as described in Figure 3, followed by the reconstruction of an image from the R, G, and B channels.

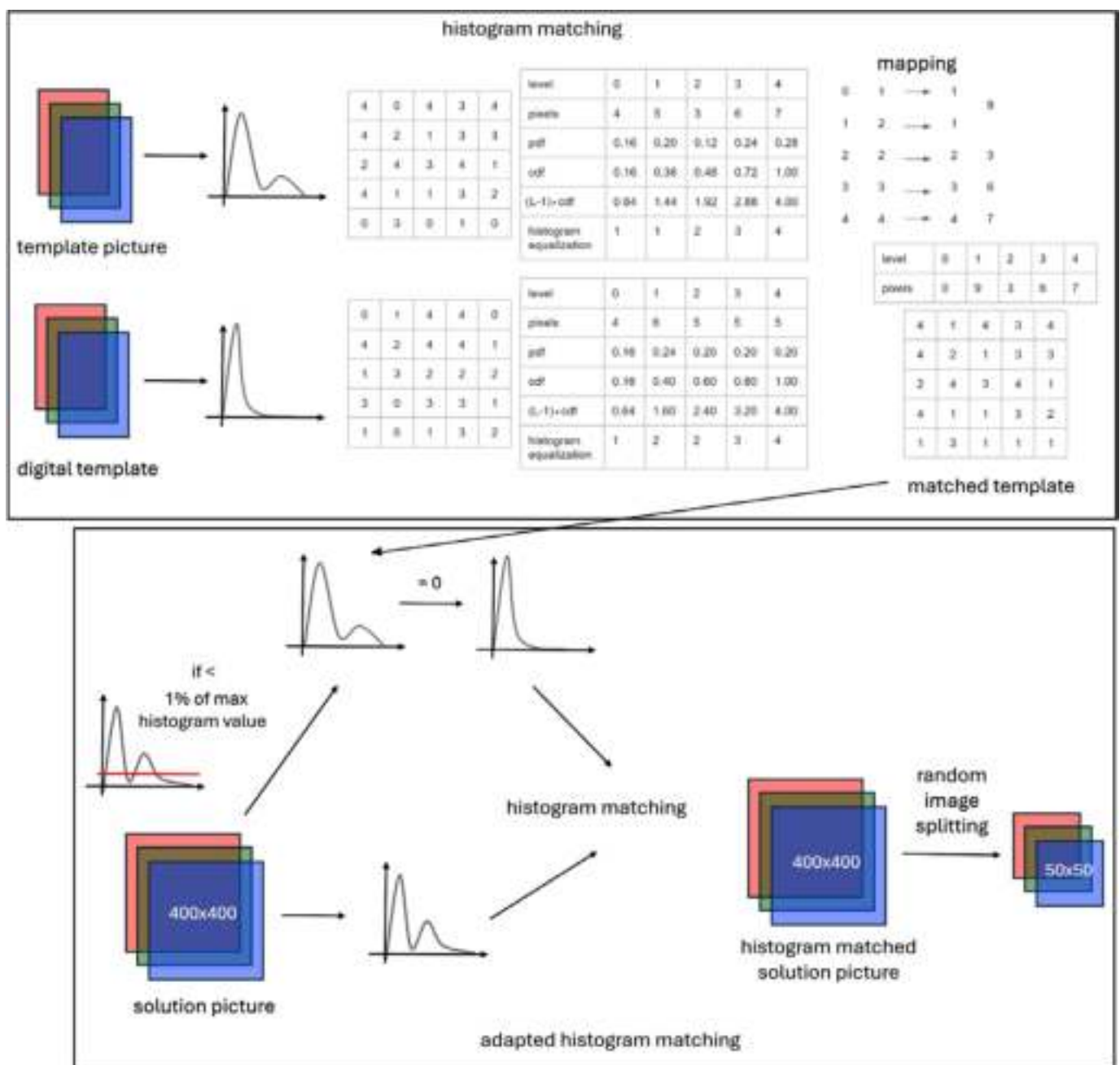


Figure 2. Diagram of the color matching algorithm.



Figure 3. Vitamin C solutions (0, 4.54, 11.31, 18.05, 22.52, 33.63, 44.64, 66.37, and 87.72 $\mu\text{g}\cdot\text{mL}^{-1}$) in the presence of double-distilled water, acetate buffer, ammonium iron(III) sulfate, and orthophenanthroline.

3.2. Software

All calculations were performed in Python 3.10.0. The following libraries and packages were used: opencv 4.8.1, scikit-image 0.22.0, numpy 1.26.1, matplotlib 3.5.3, scikit-learn 1.0.2, tensorflow 2.9.1, and keras 2.10.0.

4. Results and Discussion

4.1. Reagents, Laboratory Equipment and Measurement Procedure

In this work, the colorimetric reaction used to determine ascorbic acid (AA) was based on the reduction of Fe(III) to Fe(II) by AA, which forms a red color complex with orthophenanthroline. In this experiment, L(+)-ascorbic acid pure p.a., CAS: 50-81-7 (POCH, Gliwice, Poland), 1,10 phenanthroline p.a., CAS: 66-71-7 (POCH, Poland), and ammonium iron(III) sulfate dodecahydrate pure p.a., CAS: 7783-83-7 (POCH, Poland) were used. The 1 M acetate buffer was prepared in our laboratory by mixing 1.0 mol·L⁻¹ sodium acetate (CH₃COOH) with 1.0 mol·L⁻¹ acetic acid (CH₃COONa) (both reagents purchased from Avantor Performance Materials, Gliwice, Poland) and adjusting to the desired pH using 10N HCl. All solutions, including AA at a concentration of 2.5 gL⁻¹, phenanthroline at a concentration of 4.0 gL⁻¹, ammonium iron(III) sulfate at a concentration of 2.5 gL⁻¹, and 1 M acetate buffer (pH 4.6), were prepared with double-distilled water. Small laboratory equipment was used in the experiments, including a precise analytical balance (RADWAG, model AS 60/220.XS, Radom, Poland), a magnetic stirrer (WIGO, Toszek, Poland), automatic pipettes of various volumes (Eppendorf, Germany), and glassware, i.e., beakers and Petri dishes. To adjust the pH values of the buffer solution, we used the SevenCompact S210 laboratory pH meter (Mettler Toledo, Greifensee, Switzerland). All experiments were performed at room temperature.

Solutions with different concentrations of vitamin C for direct color assessment were prepared by first adding 9 mL of distilled water to the beakers, then 1 mL of acetate buffer, then the appropriate volume of ascorbic acid, and 0.5 mL of ammonium iron(III) sulfate. After waiting for 3 min, 0.5 mL of phenanthroline was added. The additions of AA were 0, 20, 50, 80, 100, 150, 200, 300, and 400 µL. Figure 3 shows the photo of the samples prepared for further testing. The parameters studied for optimization were reagent concentration, volume, and the time between the individual stages (the influence of chemical kinetics). After waiting about 20 min, 10 mL of each colored solution was poured into Petri dishes with flat bottoms and photos were taken with a smartphone camera.

4.2. Preparation of a Color Template and Picture Acquisition

The experiment was divided into several steps and the procedure is shown in Figure 4.

The first step was the preparation of the template for color standardization. To determine which colors should be used as a reference, a series of vitamin C solutions was prepared covering the concentration range considered in this study. This was done to estimate the range of RGB values significant for these calculations. Pictures of the tested solutions were taken, and using dedicated homemade software, 12 configurations of RGB values (colors) were chosen to be used on the template. The averaged RGB values formed the basis for determining the color range in the template dedicated to the problem being solved. To slightly expand the color palette, additional upper and lower values were estimated, and the template was created to contain 12 colors. The template, with each color represented by one square, was printed in several copies with one printer. The size of each square was 180 × 180 pixels, and the total size of the template was 600 × 790 pixels.

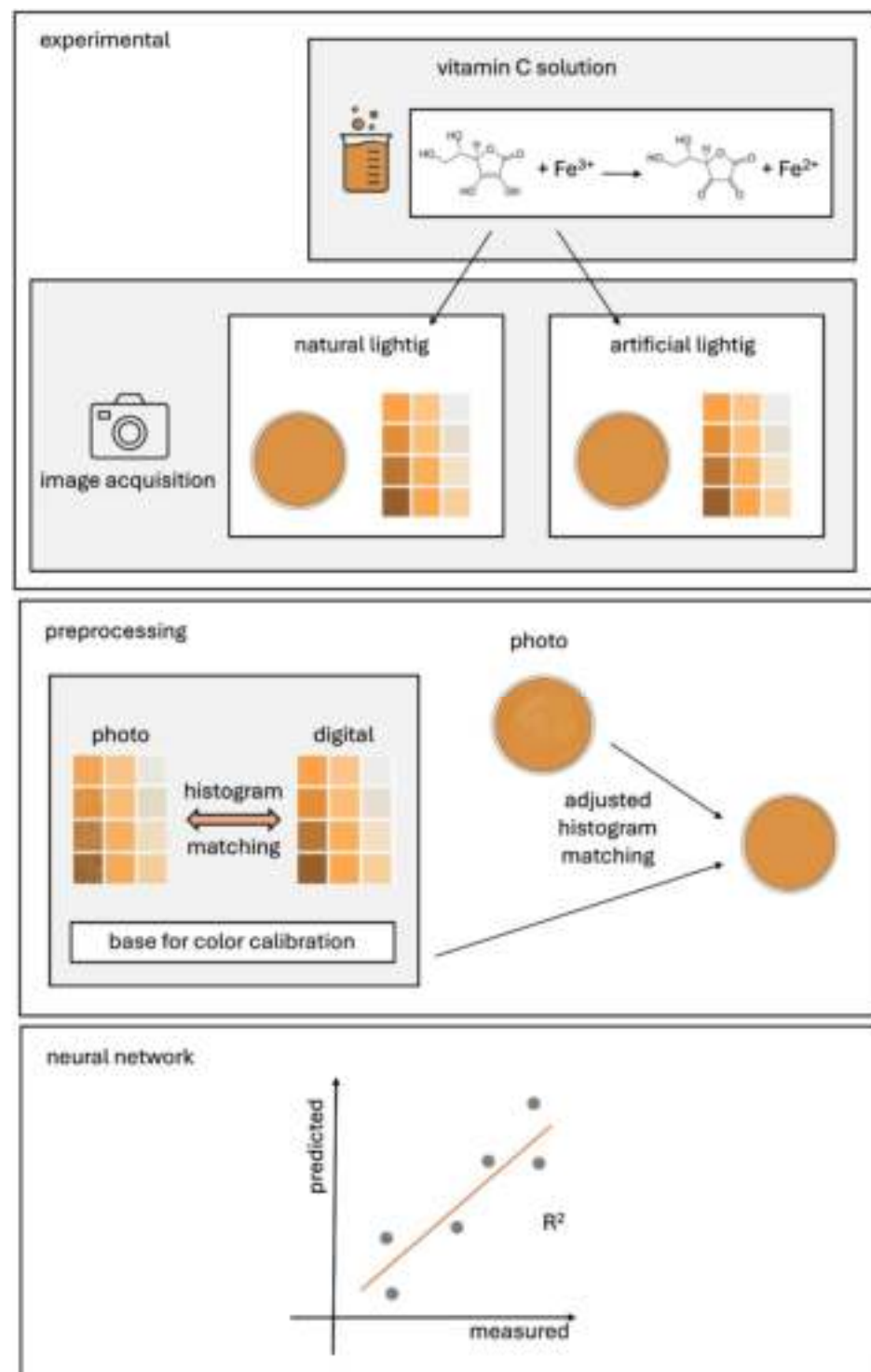


Figure 4. Scheme of conducted experiments and calculations: preparation of AA solutions with varying concentrations; application of solutions onto Petri dishes; image capture under different lighting conditions (AA solutions with color templates); image preprocessing—color standardization; training and evaluation of the regression network.

The process of taking the pictures involved placing 10 mL of the solution in a Petri dish on a white background (highlighting the contrast between the background and the Petri dish with solution), next to the template (which served as a color control set) on a black surface. There were two conditions in which the pictures were taken, natural lighting and the light from the torch of a smartphone. In the second variant, a box completely

isolating the photographed objects from the surroundings was placed onto the experimental setup to cut off access to the natural light. Differences in the colors of solutions and the photographed color charts were observed under these two illuminations (Figure 5). The process was repeated 11 times for each sample with a different concentration of vitamin C. All of the pictures were taken with a Xiaomi smartphone (Xiaomi Redmi Note 9, Xiaomi, Beijing, China), which has a 48 MP camera with an $f/1.79$ aperture. The resolution of the pictures was 3984×1840 px.

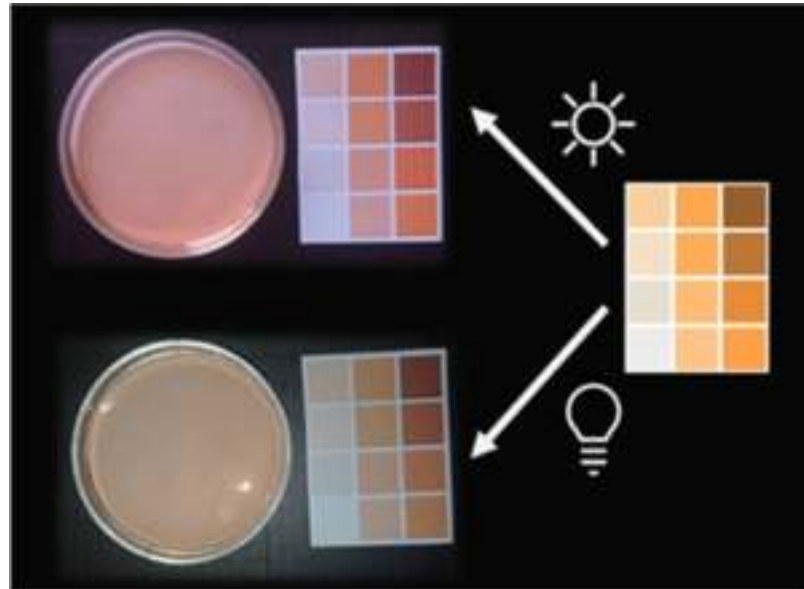


Figure 5. Control of lighting conditions—two approaches: natural light (**top**) and smartphone lamp lighting (**bottom**). Right side of an image—digital template used in histogram matching; arrows indicate the lighting condition under which the images were taken.

To histogram-match the templates used in standardization, additional pictures were taken of the template under both conditions. Templates were cut from the images, and histogram matching was performed between the resulting templates and the original one. The process with the results can be seen in Figure 6. Each template is presented as an RGB image, along with its histograms for each of the R, G, and B channels. This figure clearly illustrates the operation of our algorithm. The first column shows the ideal histogram of the digital color template designed for this task. The color histograms of the template photos are expected to match the ideal characteristics. However, under real conditions, there are significant differences (second and fourth columns) between the histograms of the photos and those of the digital version. However, after applying the algorithm proposed in this work, the expected shape of the color histogram for all three components, R, G, and B, can be reproduced. The graphs in the third and fifth columns match those in the first column.

The cumulative histograms of the matched images are the same as the cumulative histogram of the digital template (which serves as the reference) for each channel. Each spike on a given histogram corresponds to one component of a certain color on the template. The less discrete appearance of the histograms of the original template images is caused by variance from printing and different variables that affect how the camera perceives the object with certain changes (e.g., shadows and highlights).

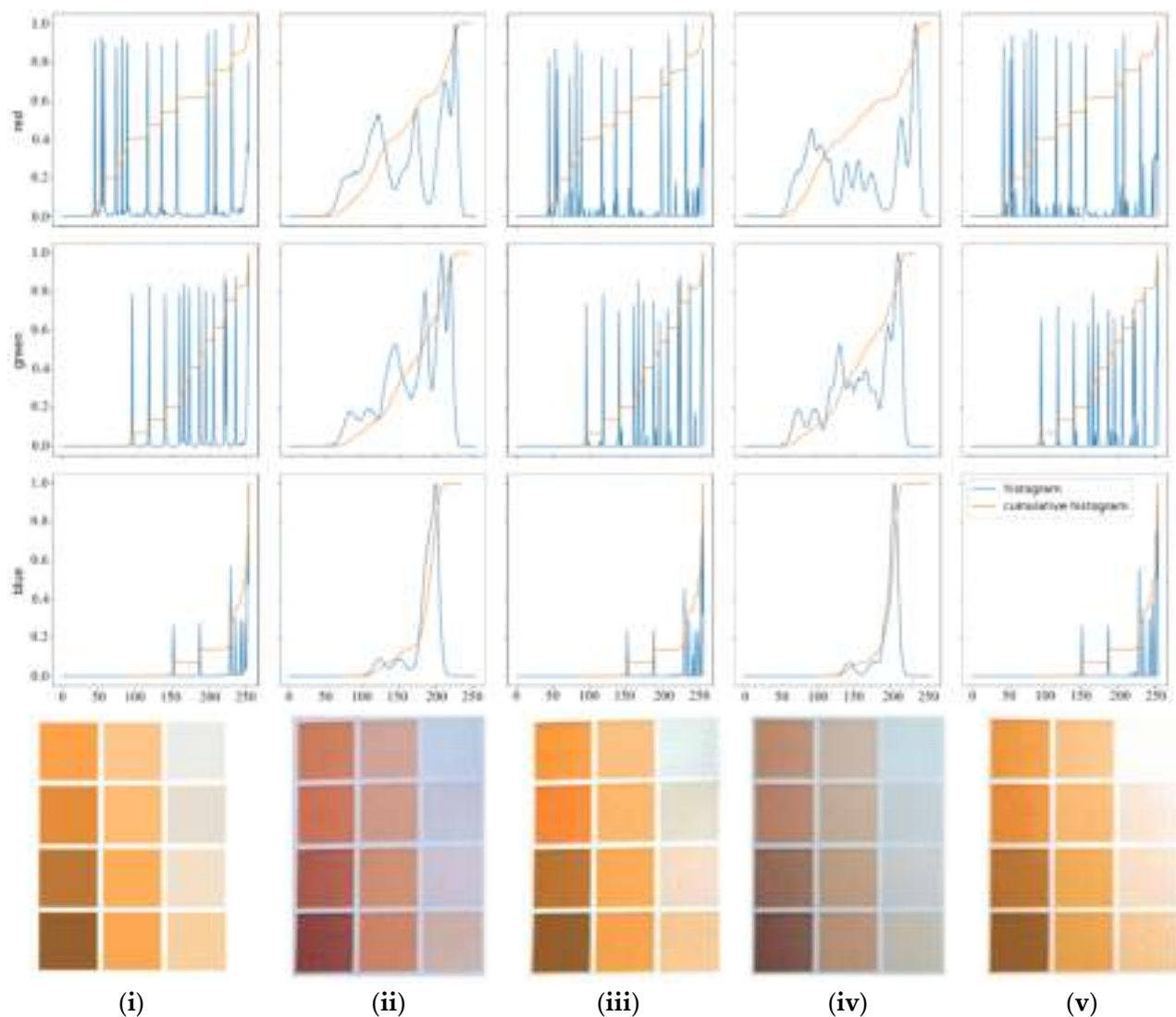


Figure 6. Templates and their histograms for RGB channels: (i) original digital template, (ii) image of a template under natural lighting, (iii) image of a matched template under natural lighting, (iv) image of a template under artificial lighting, and (v) image of a matched template under artificial lighting.

The dataset for the calculations was prepared as follows: 198 pictures were taken for nine concentrations of vitamin C under two different lighting conditions. This means that for a given solution for each of the conditions, 11 pictures were taken. Each of the images underwent a process of standardization, as described in Section 3.1. The exemplary effect is illustrated in Figure 7. In Figure 8, the effect of applying the algorithm to various images of the vitamin C solution with a concentration of $33.63 \mu\text{g/mL}$ is presented. This resulted in the generation of datasets for two lighting conditions. The combination of both sets resulted in the preparation of a mixed condition set, which consisted of:

- training set: 60,416 images
- validation set: 20,480 images
- test set: 20,480 images.

The prepared image datasets served as input data for training a regression neural network, which was used to obtain a model for determining AA concentration based on the photos taken.

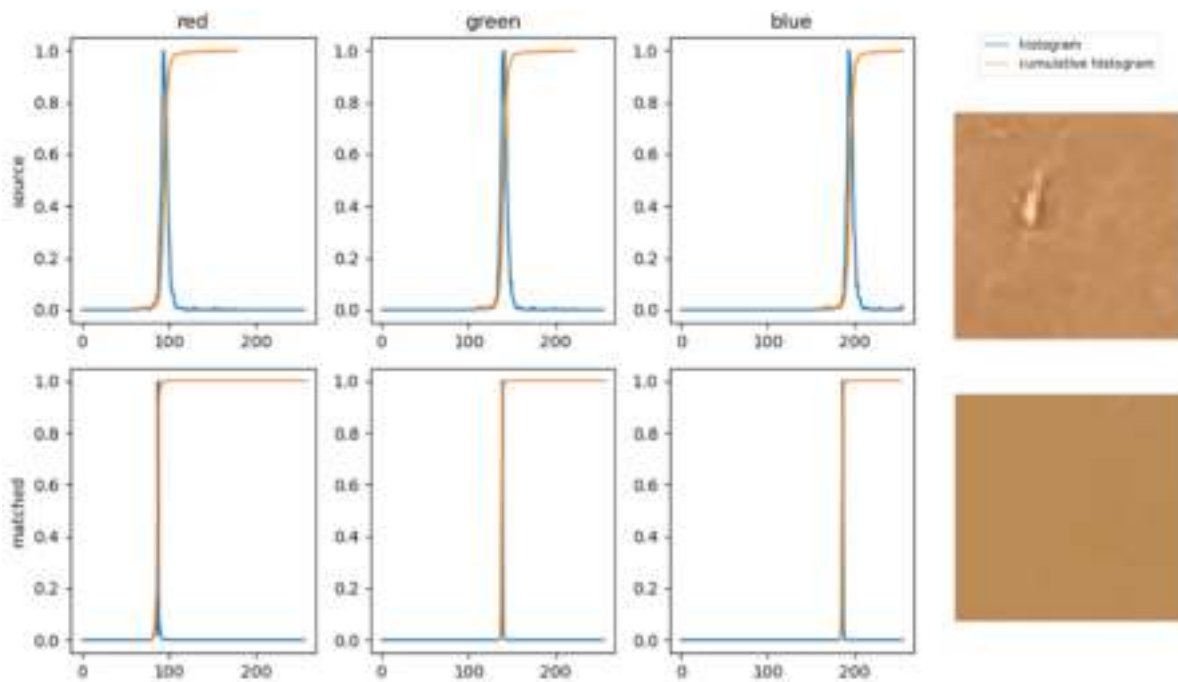


Figure 7. Images of the vitamin C solution (33.63 $\mu\text{g/mL}$) under artificial lighting conditions before (source) and after (matched) application of the standardization algorithm and corresponding histograms of RGB channels.

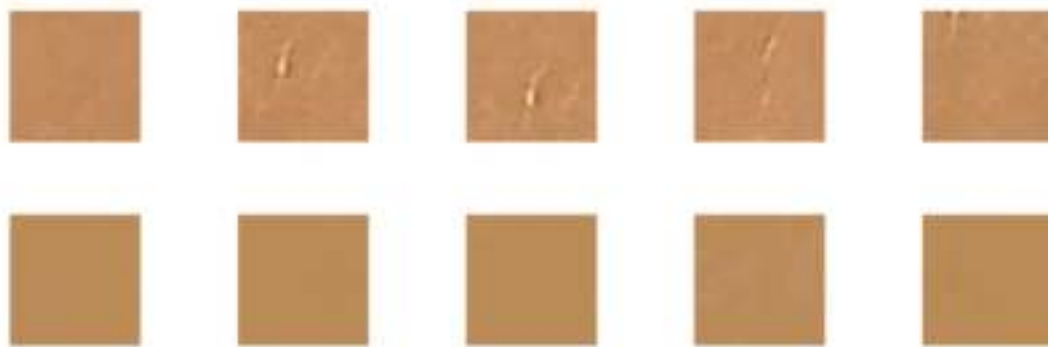


Figure 8. Various exemplary images of the vitamin C solution (33.63 $\mu\text{g/mL}$) under artificial lighting conditions before (**top**) and after (**bottom**) application of the standardization algorithm.

4.3. Network Architecture

A multivariate regression model was defined using deep neural networks with an unconventional architecture specifically dedicated to this task. The network design considered essential information such as color-based modeling, the similarity of colors corresponding to successive values of the target variable, and the approximately uniform color of each photo.

The neural network's input layer accepted images with dimensions of 50×50 and three color channels (RGB), randomly selected from each photo. Subsequently, Conv2D convolutional layers with small 1×1 filters, L2 regularization, and the GELU (Gaussian error linear unit) activation function were applied after each convolutional layer. A normalization layer was also used, operating along the feature axis for each sample to stabilize and improve the learning process. The architecture also included pooling layers that reduced the spatial size of the feature map, including AveragePooling2D and GlobalMaxPooling2D, which reduced the spatial dimension to one value per feature map. The combination of global pooling and flattening is a standard technique for dimension reduction before fully connected layers. In the next stage, the data was flattened to a 1D vector, which was passed

to fully connected dense layers with GELU activation and L2 regularization, while the addition of a Dropout layer prevented overfitting. The output was a value corresponding to the predicted concentration of the solution in the image. A detailed description of the network architecture is provided in Table 2 and an overview is illustrated in Figure 9. The total number of parameters was 23,879, and training in a single cycle took a few to several seconds (depending on the computation variant).

Table 2. Neural network architecture and parameters used in this study.

Layer (Type)	Output Shape	Number of Parameters
Input layer	(None, 50, 50, 3)	0
Conv2D	(None, 50, 50, 3, 2)	128
Activation	(None, 50, 50, 32)	0
Conv2D	(None, 50, 50, 64)	2112
Activation	(None, 50, 50, 64)	0
AveragePooling2D	(None, 46, 46, 64)	0
LayerNormalization	(None, 46, 46, 64)	128
Conv2D	(None, 46, 46, 64)	4160
Activation	(None, 46, 46, 64)	0
AveragePooling2D	(None, 44, 44, 64)	0
GlobalMaxPooling2D	(None, 64)	0
Flatten	(None, 64)	0
Dense	(None, 150)	9750
Dropout	(None, 150)	0
Dense	(None, 50)	7550
Dense	(None, 1)	51
Total params: 23,879 (93.28 KB)		
Trainable params: 23,879 (93.28 KB)		
Non-trainable params: 0 (0.00 Byte)		

In a convolutional layer such as Conv2D, the applied L2 regularization works on the convolutional filters (kernels). These filters are responsible for extracting features from the input images. Adding L2 regularization means that during the optimization process, the loss function the model tries to minimize includes an additional term that penalizes large weight values of the filters. Adding L2 regularization to the convolutional layer helps prevent overfitting, stabilizes the optimization process, and promotes more uniform and stable solutions, leading to better model generalization.

The kernel size of 1×1 in the convolutional layer has specific and useful properties in neural networks. The 1×1 filters are typically used to reduce spatial dimensions without spatial aggregation, which can be useful in reducing the number of feature channels or introducing nonlinearity without changing the spatial size. This configuration is often called a “pointwise convolution”. The 1×1 kernel changes the number of output channels (depth) without changing the width and height of the input image, allowing for the combination of information contained in different input channels without integrating spatial information. Each output point is a linear combination of the input channel values at a given spatial point. For example, if at a given input point (x, y) we have values in the channels (w_1, w_2, w_3) , the 1×1 kernel applies different weights to these values and combines them, creating new output channels. Such a kernel has fewer parameters compared to larger convolutional kernels, resulting in fewer computational operations. This is efficient in terms of memory and computation time. In modeling based on RGB components, using a 1×1 kernel means examining the relationships between individual color components,

rather than the neighborhood of each pixel. In our task, there was no need to detect small and then larger details, as is the case in shape recognition.

The GELU activation function [47], applied as an alternative to ReLU (Rectified Linear Unit) and other activation functions in neural networks, ensures better gradient behavior and more efficient learning in deep neural networks (Figure 10). Its continuity, differentiability, and ability to preserve nonlinear data characteristics make it a valuable tool in the design of deep learning models. Unlike ReLU, which can suffer from gradient flow issues during training (known as the vanishing gradient problem), GELU provides better gradient throughput due to its structure. GELU preserves nonlinear features that are crucial for learning data representations, helping models better capture complex dependencies in training data.



Figure 9. Neural network architecture used in this study.

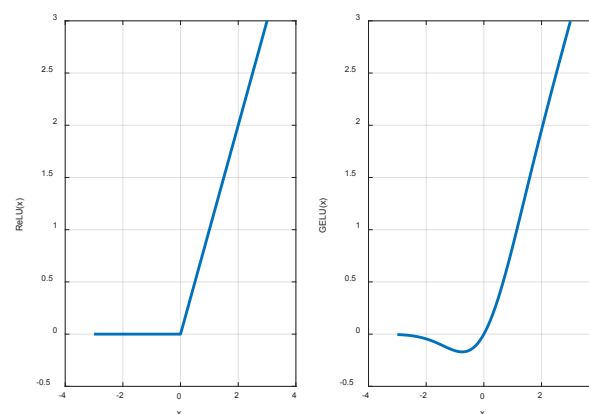


Figure 10. ReLU and GELU activation functions.

In the LayerNormalization layer, normalization is performed along the feature axis for each sample, meaning that normalization is independent of other samples in the batch. Each feature vector (e.g., channels in an image) was normalized separately, considering only the values within that vector, not the entire batch. Unlike BatchNormalization, which operates depending on batch size, LayerNormalization remains stable even with small batches or a batch size of one. Normalization can help prevent issues like vanishing or exploding gradients and accelerate convergence. Normalizing the activations helps maintain values within a reasonable range, stabilizing the learning process. When activations are normalized, the optimizer (learning algorithm) finds it easier to adjust network weights, speeding up convergence to optimal weight values.

GlobalMaxPool2D is a layer used in neural networks to reduce spatial data dimensions after applying convolutional layers or other layers processing spatial data, such as pooling layers [48,49]. GlobalMaxPool2D operates by selecting the maximum value of each feature channel across the entire spatial area (all points) of the input feature tensor. GlobalMaxPool2D is an effective tool for spatial dimensionality reduction, retaining only the most significant features (highest values) from each feature channel. This reduces the number of model parameters, which can help reduce overfitting and computational complexity. GlobalMaxPool2D has no trainable parameters; its sole purpose is to select the maximum value from each feature channel throughout the feature map. For each feature channel, the GlobalMaxPool2D layer selects the highest value across the entire spatial area (height and width). The result of GlobalMaxPool2D is a tensor with reduced dimensions, containing only the maximum values for each feature channel. After applying GlobalMaxPool2D, the resulting tensor can be passed to a fully connected layer or other types of layers that process one-dimensional data vectors.

In this study, the model was compiled using the Adam optimizer with customized beta parameters and a defined loss function, which was the sum of MSE and MAE. When the loss function is a combination of MSE and MAE, it affects how the model learns to minimize errors. MSE penalizes large errors more severely (due to the squares of differences), whereas MAE treats all errors equally. Combining these loss functions means that the model will aim to reduce both large and small errors. The gradient of MSE is steeper for larger errors, causing the model to learn faster from significant errors. The MAE gradient is constant, leading to a more consistent learning rate across all errors. MAE is more resistant to outliers than MSE because it does not involve squared differences. A loss function that combines MSE and MAE can lead to faster convergence in some cases by leveraging the strengths of both methods. MSE can accelerate the reduction of large errors, while MAE can provide stability and a consistent learning pace.

During training, a modification of the learning rate was applied after reaching a stable val_loss level for five epochs. The initial value was 0.001. Details regarding the learning parameters are provided in Table 3.

Table 3. Parameters of network training.

Parameter	Setting
Image size	50 × 50 × 3
Loss function	MSE + MAE
Optimizer	Adam
Initial learning rate	0.001
Metric	RMSE
Batch size	30
Epoch	20
Shuffle	Every epoch

The input dataset consisted of images with dimensions of 400 × 400 and was divided into three parts: training, validation, and test sets in a 60/20/20 ratio. During training,

50×50 squares were dynamically cropped from the images, and these squares served as the direct input to the first layer of the network.

There were six different variations of testing in deep learning for training a regression algorithm to determine vitamin C concentration based on images: original images for artificial lighting, matched images for artificial lighting, original images for natural lighting, matched images for artificial lighting, original images for mixed dataset, and matched images for mixed dataset. For each of them, the model was trained (Section 3.2) and predictions were made. Training and validation losses for calculations of artificial light data, for both original and histogram-matched images, are presented in Figure 11. The range was similar in both cases; however, the value fluctuations were smaller for matched images, and in this condition, validation and training losses were more similar to each other.

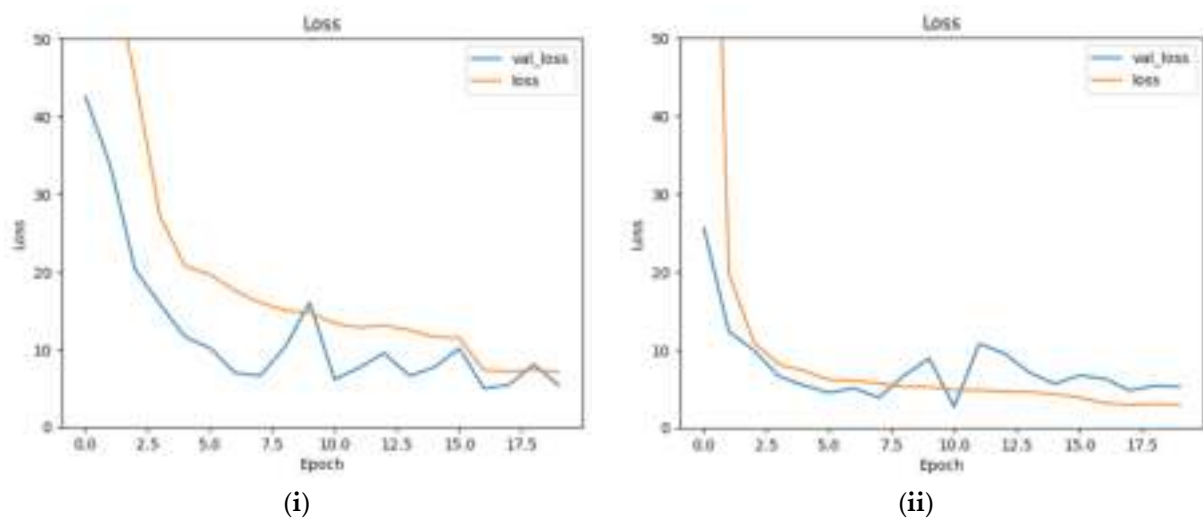


Figure 11. Training and validation losses for calculations of artificial lightning conditions for both original and (i) and histogram-matched (ii) images.

Figure 12 presents the results of the prediction made using the presented neural network. There were three sets of pairs of outcomes: artificial lightning, natural lightning, and mixed data. Each set was repeated for both the original images and the processed ones. There was an improvement in the higher concentration values for each of the conditions (Table 4). The standard deviation of the predictions was lower when applying the proposed standardization method. The highest difference could be seen in the case of artificial light, which could be caused by the uniformization of light reflections visible in the original images.

Table 4. Evaluation of DL regression models with the best results for six different variations of testing.

	Artificial Lighting, Original	Artificial Lighting, Matched	Natural Lighting, Original	Natural Lighting, Matched	Mixed Lighting, Original	Mixed Lighting, Matched
Slope	0.9799	0.9961	0.9706	0.9452	0.9784	0.9825
Intercept	−0.0036	−0.2091	1.1888	0.6051	0.5832	0.4086
RMSE/ $\mu\text{g}\cdot\text{mL}^{-1}$	1.5769	0.7525	2.2900	1.9542	1.4828	1.3510
R ² score	0.9979	0.9999	0.9953	0.9956	0.9966	0.9968

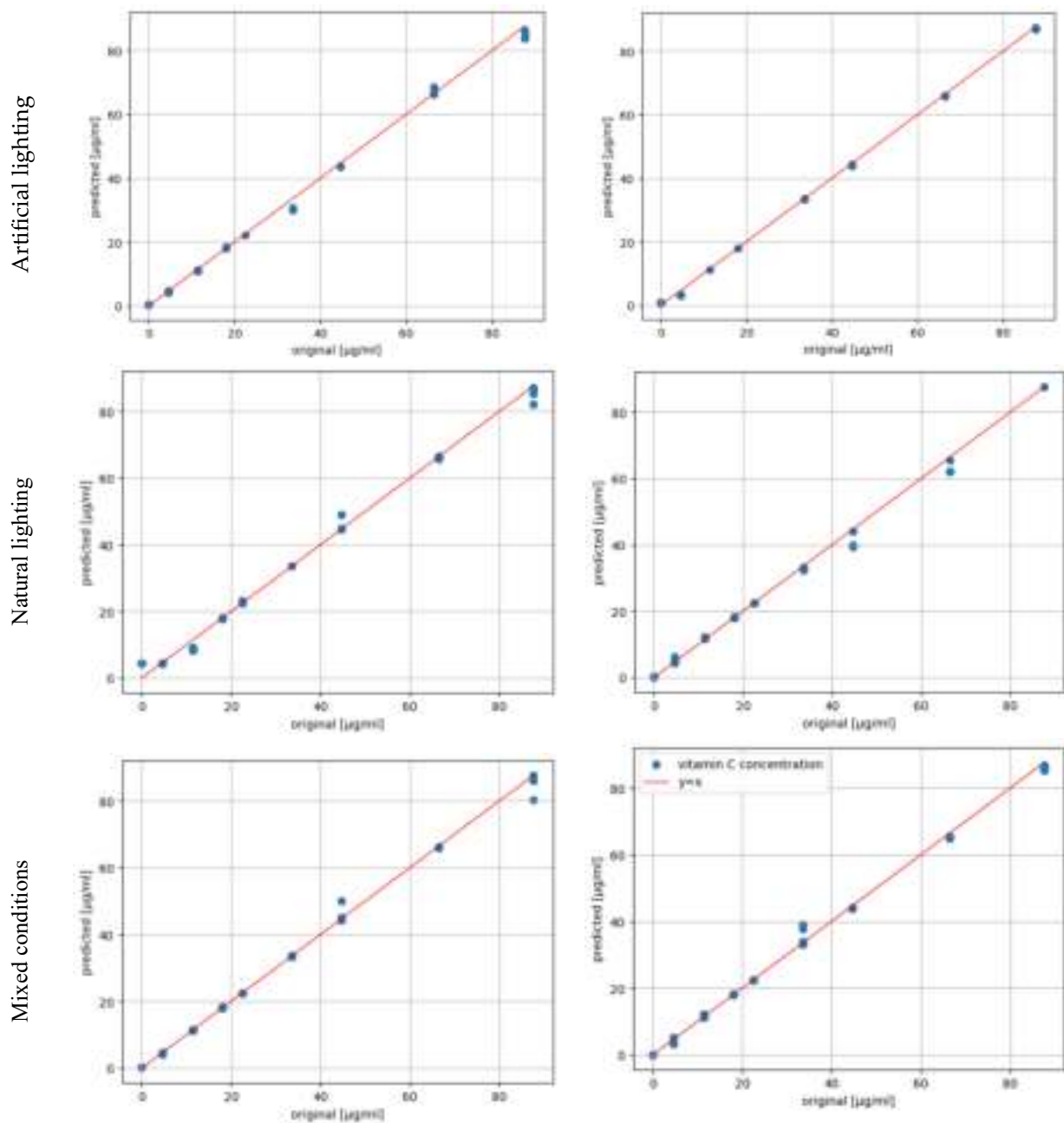


Figure 12. Evaluation of DL regression models for six different variations of testing, before (first column) and after color matching (second column).

The values of the numerical model evaluation and adjustment factors are presented in Table 4. The R^2 score was very high (above 0.99) in all cases; however, in every instance, the R^2 score was higher when standardization was performed. As for root mean squared error values, they were lower for matched images compared to the original ones. Depending on the lighting conditions, the learning rate changed from an initial value of 10^{-3} to 10^{-5} .

4.4. Ablation Study

In this study, we also analyzed how different modifications of the network architecture affected the quality of the regression model. In the first stage, we considered a network that did not account for the specifics of the problem. The model contained six layers: one convolution, one pooling, one flatten, and three dense. It was trained under the same conditions as earlier. The detailed quantitative results are presented in Table 5. The use

of a more conventional architecture worsened the model's ability to fit the data and its predictive capabilities. However, improvements in modeling quality were still observed after applying the color standardization algorithm.

Table 5. Evaluation of DL regression models considered in the ablation study for six different variations of testing, depending on the lighting conditions and chosen dataset.

	Artificial Lighting, Original	Artificial Lighting, Matched	Natural Lighting, Original	Natural Lighting, Matched	Mixed Lighting, Original	Mixed Lighting, Matched
Slope	0.9058	1.0138	0.9538	1.0071	1.0005	1.0112
Intercept	0.6891	−0.3598	−0.8347	0.2263	0.8747	0.0203
RMSE/ $\mu\text{g}\cdot\text{mL}^{-1}$	5.8415	1.7077	3.9504	2.5151	4.3488	2.8211
R ² score	0.9557	0.9962	0.9798	0.9918	0.9755	0.9897

Moreover, several various changes in the architecture were introduced before the final version was chosen. Some of them, with the comparison to the final version, are evaluated in Table 6.

Table 6. Evaluation of DL regression models considered in the ablation study for mixed lighting and matched condition of testing, depending on changes in the model.

Change in Architecture	RMSE/ $\mu\text{g}\cdot\text{mL}^{-1}$	R ² Score	Comparison
Batch size = 120	2.3535	0.9947	Decrease of R ² score and increase of RMSE in comparison to final model.
Image size = 16	4.0146	0.9825	Significant decrease of R ² score and increase of RMSE in comparison to final model. Lower decline of initial rate in comparison to final version.
Loss function modification	3.8203	0.9856	Decrease of R ² score and significant increase of RMSE in comparison to final model. Minimized differences between training and validation loss during training.
Optimizer = Adagrad	3.1625	0.9877	Decrease of R ² score and increase of RMSE in comparison to final model.
Optimizer = SGD	-	-	There was no change in loss value between the epochs. The model did not learn to predict the concentration of vitamin C solutions.
Initial learning rate = 0.01 or 0.0001	-	-	No significant change in the results was detected. The results achieved did not vary significantly in comparison to final model results.

5. Conclusions

This work presents a new color standardization algorithm, in which the main stages consist of preparing a template based on an initial series of photos, developing a color mapping principle using equalized digital RGB histograms of an ideal pattern and its photo, color conversion of photographs of the examined object involving the use of information about mapping template histograms, and image reconstruction based on the transformed histogram. The algorithm was developed for cases where the examined objects differ little in color and the color of each photographed object is almost uniform. The operation of the proposed procedure was demonstrated and verified in an analytical chemistry task, which involved the colorimetric determination of vitamin C. Photographs of AA solutions taken under controlled conditions did not show much color difference within the concentration range. The research was carried out under daylight and artificial lighting conditions

in order to combine them into a single dataset and obtain a model that allows for the determination of vitamin C regardless of the lighting conditions used when taking the photos. Sets of standardized images formed the basis for deep machine learning using the architecture of neural networks in the regression variant.

For the concentration range of vitamin C from 0 to $87.72 \mu\text{g}\cdot\text{mL}^{-1}$, the RMSE for the test set ranged between 0.75 and $1.95 \mu\text{g}\cdot\text{mL}^{-1}$ in comparison to the non-standardized variant, where the RMSE was between 1.35 and $2.29 \mu\text{g}\cdot\text{mL}^{-1}$. The compatibility of the concentration prediction results with the actual data, as expressed by R^2 , was greater than 0.99 for each of the standardized variants. The greatest improvement was noted in the case of artificial light, which can be explained by the elimination of reflections on the surfaces of the solutions in the images after the standardization process. Further studies are needed to determine the usefulness of the algorithm in different scenarios (e.g., colorimetric prediction of the concentration of different compounds). This new approach may contribute to the development of a new field of chemical analysis, which includes the quantitative evaluation of a broad spectrum of chemical analytes using mobile phones and image analysis. This presented study shows promise for applications in several fields of science and technology, such as biomedical engineering, chemical engineering, and the food industry.

Author Contributions: P.K.: Conceptualization; Methodology; Data curation; Formal analysis; Investigation; Software; Resources; Visualization; Writing—original draft; Writing—Review and Editing. M.J.: Conceptualization; Formal analysis; Funding acquisition; Investigation; Methodology; Project administration; Supervision; Validation; Writing—original draft; Writing—Review and Editing. All authors have read and agreed to the published version of the manuscript.

Funding: Research project supported by the program “Excellence initiative—research university” for the AGH University of Krakow (ID1479).

Data Availability Statement: The algorithms implemented in Python and the sample of the data for calculations are available on the Github platform at: <https://github.com/pwkwiek/vitaminC> (accessed on: 13 July 2024).

Acknowledgments: The authors thank Inż. Filip Ciepiela (AGH University of Krakow) for valuable guidance during the project implementation.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Fairchild, M.D. *Color Appearance Models*, 3rd ed.; John Wiley & Sons, Ltd.: Hoboken, NJ, USA, 2013; ISBN 9781119967033.
2. Minz, P.S.; Saini, C.S. Evaluation of RGB Cube Calibration Framework and Effect of Calibration Charts on Color Measurement of Mozzarella Cheese. *J. Food Meas. Charact.* **2019**, *13*, 1537–1546. [\[CrossRef\]](#)
3. Ernst, A.; Papst, A.; Ruf, T.; Garbas, J.U. Check My Chart: A Robust Color Chart Tracker for Colorimetric Camera Calibration. In Proceedings of the 6th International Conference on Computer Vision/Computer Graphics Collaboration Techniques and Applications, MIRAGE’13, Berlin, Germany, 6–7 June 2013.
4. McCamy, C.S.; Marcus, H.; Davidson, J.G. Color-Rendition Chart. *J. Appl. Photogr. Eng.* **1976**, *2*, 95–99.
5. Sunoj, S.; Igathinathane, C.; Saliendra, N.; Hendrickson, J.; Archer, D. Color Calibration of Digital Images for Agriculture and Other Applications. *ISPRS J. Photogramm. Remote Sens.* **2018**, *146*, 221–234. [\[CrossRef\]](#)
6. Kim, M.; Kim, B.; Park, B.; Lee, M.; Won, Y.; Kim, C.Y.; Lee, S. A Digital Shade-Matching Device for Dental Color Determination Using the Support Vector Machine Algorithm. *Sensors* **2018**, *18*, 3051. [\[CrossRef\]](#)
7. Karaimer, H.C.; Brown, M.S. Improving Color Reproduction Accuracy on Cameras. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Salt Lake, UT, USA, 18–23 June 2018; pp. 6440–6449.
8. Roy, S.; Kumar Jain, A.; Lal, S.; Kini, J. A Study about Color Normalization Methods for Histopathology Images. *Micron* **2018**, *114*, 42–61. [\[CrossRef\]](#)
9. Zhao, Y.; Ferguson, S.; Zhou, H.; Elliott, C.; Rafferty, K. Color Alignment for Relative Color Constancy via Non-Standard References. *IEEE Trans. Image Process.* **2022**, *31*, 6591–6604. [\[CrossRef\]](#)
10. Rashid, F.; Jamayet, N.B.; Farook, T.H.; AL-Rawas, M.; Barman, A.; Johari, Y.; Noorani, T.Y.; Abdullah, J.Y.; Eusufzai, S.Z.; Alam, M.K. Color Variations during Digital Imaging of Facial Prostheses Subjected to Unfiltered Ambient Light and Image Calibration Techniques within Dental Clinics: An In Vitro Analysis. *PLoS ONE* **2022**, *17*, e0273029. [\[CrossRef\]](#)
11. Barbero-Álvarez, M.A.; Rodrigo, J.A.; Menéndez, J.M. Minimum Error Adaptive RGB Calibration in a Context of Colorimetric Uncertainty for Cultural Heritage Preservation. *Comput. Vis. Image Underst.* **2023**, *237*, 103835. [\[CrossRef\]](#)

12. Noor Azhar, M.; Bustam, A.; Naseem, F.S.; Shuin, S.S.; Md Yusuf, M.H.; Hishamudin, N.U.; Poh, K. Improving the Reliability of Smartphone-Based Urine Colorimetry Using a Colour Card Calibration Method. *Digit. Health* **2023**, *9*, 20552076231154684. [\[CrossRef\]](#)
13. Zhang, G.; Song, S.; Panescu, J.; Shapiro, N.; Dannemiller, K.C.; Qin, R. A Novel Systems Solution for Accurate Colorimetric Measurement through Smartphone-Based Augmented Reality. *PLoS ONE* **2023**, *18*, e0287099. [\[CrossRef\]](#)
14. Chairat, S.; Chaichulee, S.; Dissaneewate, T.; Wangkulangkul, P.; Kongpanichakul, L. AI-Assisted Assessment of Wound Tissue with Automatic Color and Measurement Calibration on Images Taken with a Smartphone. *Healthcare* **2023**, *11*, 273. [\[CrossRef\]](#)
15. Suominen, J.; Egiazarian, K. Camera Color Correction Using Splines. In Proceedings of the IS&T International Symposium on Electronic Imaging, Burlingame, CA, USA, 21–25 January 2024; pp. 165–1–165–6.
16. Souissi, M.; Chaouch, S.; Moussa, A. Color Matching of Bicomponent (PET/PTT) Filaments with High Performances Using Genetic Algorithm. *Sci. Rep.* **2024**, *14*, 10949. [\[CrossRef\]](#)
17. Wannasin, D.; Grossmann, L.; McClements, D.J. Optimizing the Appearance of Plant-Based Foods Using Natural Pigments and Color Matching Theory. *Food Biophys.* **2024**, *19*, 120–130. [\[CrossRef\]](#)
18. Wu, Y. Reference Image Aided Color Matching Design Based on Interactive Genetic Algorithm. *J. Electr. Syst.* **2024**, *20*, 400–410. [\[CrossRef\]](#)
19. Food and Agriculture Organization; World Health Organization. *Vitamin and Mineral Requirements in Human Nutrition*, 2nd ed.; FAO/WHO: Geneva, Switzerland, 1998; pp. 1–20, ISBN 9241546123.
20. Food and Agriculture Organization; World Health Organization. *Human Vitamin and Mineral Requirements*; FAO/WHO: Geneva, Switzerland, 2001.
21. Lykkesfeldt, J. On the Effect of Vitamin C Intake on Human Health: How to (Mis)Interpret the Clinical Evidence. *Redox Biol.* **2020**, *34*, 101532. [\[CrossRef\]](#)
22. Dosed, M.; Jirkovsk, E.; Kujovsk, L.; Javorsk, L.; Pourrov, J.; Mercolini, L.; Remi, F. Vitamin C—Sources, Physiological Role, Kinetics, Deficiency, Use, Toxicity, and Determination. *Nutrients* **2021**, *615*, 1–34.
23. Suntornsuk, L.; Gritsanapun, W.; Nilkamhank, S.; Paochom, A. Quantitation of Vitamin C Content in Herbal Juice Using Direct Titration. *J. Pharm. Biomed. Anal.* **2002**, *28*, 849–855. [\[CrossRef\]](#)
24. Klimczak, I.; Gliszczynska-Świąło, A. Comparison of UPLC and HPLC Methods for Determination of Vitamin C. *Food Chem.* **2015**, *175*, 100–105. [\[CrossRef\]](#)
25. Gazdik, Z.; Zitka, O.; Petrlova, J.; Adam, V.; Zehnalek, J.; Horna, A.; Reznicek, V.; Beklova, M.; Kizek, R. Determination of Vitamin C (Ascorbic Acid) Using High Performance Liquid Chromatography Coupled with Electrochemical Detection. *Sensors* **2008**, *8*, 7097–7112. [\[CrossRef\]](#)
26. Bunaciu, A.A.; Bacalum, E.; Aboul-Enein, H.Y.; Udristioiu, G.E.; Fleschin, S. FT-IR Spectrophotometric Analysis of Ascorbic Acid and Biotin and Their Pharmaceutical Formulations. *Anal. Lett.* **2009**, *42*, 1321–1327. [\[CrossRef\]](#)
27. Zhu, Q.; Dong, D.; Zheng, X.; Song, H.; Zhao, X.; Chen, H.; Chen, X. Chemiluminescence Determination of Ascorbic Acid Using Graphene Oxide@copper-Based Metal-Organic Frameworks as a Catalyst. *RSC Adv.* **2016**, *6*, 25047–25055. [\[CrossRef\]](#)
28. Berg, R.W. Investigation of L (+)-Ascorbic Acid with Raman Spectroscopy in Visible and UV Light. *Appl. Spectrosc. Rev.* **2015**, *50*, 193–239. [\[CrossRef\]](#)
29. Yang, H.; Irudayaraj, J. Rapid Determination of Vitamin C by NIR, MIR and FT-Raman Techniques. *J. Pharm. Pharmacol.* **2010**, *54*, 1247–1255. [\[CrossRef\]](#)
30. Zykova, E.V.; Sandetskaya, N.G.; Ostrovskii, O.V.; Verovskii, V.E. Methods of Analysis and Process Control Determining Ascorbic Acid in Medicinal Preparations By Capillary Zone Electrophoresis and Micellar. *Pharm. Chem. J.* **2010**, *44*, 463–465. [\[CrossRef\]](#)
31. Dodevska, T.; Hadzhiev, D.; Shterev, I. A Review on Electrochemical Microsensors for Ascorbic Acid Detection: Clinical, Pharmaceutical, and Food Safety Applications. *Micromachines* **2023**, *14*, 41. [\[CrossRef\]](#)
32. Huang, L.; Tian, S.; Zhao, W.; Liu, K.; Guo, J. Electrochemical Vitamin Sensors: A Critical Review. *Talanta* **2021**, *222*, 121645. [\[CrossRef\]](#)
33. Broncová, G.; Prokopec, V.; Shishkanova, T.V. Potentiometric Electronic Tongue for Pharmaceutical Analytics: Determination of Ascorbic Acid Based on Electropolymerized Films. *Chemosensors* **2021**, *9*, 110. [\[CrossRef\]](#)
34. Coutinho, M.S.; Morais, C.L.M.; Neves, A.C.O.; Menezes, F.G.; Lima, K.M.G. Colorimetric Determination of Ascorbic Acid Based on Its Interfering Effect in the Enzymatic Analysis of Glucose: An Approach Using Smartphone Image Analysis. *J. Braz. Chem. Soc.* **2017**, *28*, 2500–2505. [\[CrossRef\]](#)
35. Porto, I.S.A.; Santos Neto, J.H.; dos Santos, L.O.; Gomes, A.A.; Ferreira, S.L.C. Determination of Ascorbic Acid in Natural Fruit Juices Using Digital Image Colorimetry. *Microchem. J.* **2019**, *149*, 104031. [\[CrossRef\]](#)
36. Kong, L.; Gan, Y.; Liang, T.; Zhong, L.; Pan, Y.; Kirsanov, D.; Legin, A.; Wan, H.; Wang, P. A Novel Smartphone-Based CD-Spectrometer for High Sensitive and Cost-Effective Colorimetric Detection of Ascorbic Acid. *Anal. Chim. Acta* **2020**, *1093*, 150–159. [\[CrossRef\]](#)
37. Li, C.; Xu, X.; Wang, F.; Zhao, Y.; Shi, Y.; Zhao, X.; Liu, J. Portable Smartphone Platform Integrated with Paper Strip-Assisted Fluorescence Sensor for Ultrasensitive and Visual Quantitation of Ascorbic Acid. *Food Chem.* **2023**, *402*, 134222. [\[CrossRef\]](#)
38. Zhaoa, W.; Caa, P.; Zhua, Y.; Liua, S.; Gaob, H.-W.; Huang, C. Rapid Detection of Vitamin C Content in Fruits and Vegetables Using a Digital Camera and Color Reaction. *Quim. Nov.* **2020**, *43*, 1421–1430. [\[CrossRef\]](#)

39. Dumancas, G.G.; Ramasahayam, S.; Bello, G.; Hughes, J.; Kramer, R. Chemometric Regression Techniques as Emerging, Powerful Tools in Genetic Association Studies. *TrAC Trends Anal. Chem.* **2015**, *74*, 79–88. [[CrossRef](#)]
40. Wold, S.; Sjöström, M.; Eriksson, L. PLS-Regression: A Basic Tool of Chemometrics. *Chemom. Intell. Lab. Syst.* **2001**, *58*, 109–130. [[CrossRef](#)]
41. Li, B.; Morris, J.; Martin, E.B. Model Selection for Partial Least Squares Regression. *Chemom. Intell. Lab. Syst.* **2002**, *64*, 79–89. [[CrossRef](#)]
42. Smola, A.J.; Schölkopf, B. A Tutorial on Support Vector Regression. *Stat. Comput.* **2004**, *14*, 199–222. [[CrossRef](#)]
43. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016; ISBN 9780262035613.
44. Lathuiliere, S.; Mesejo, P.; Alameda-Pineda, X.; Horaud, R. A Comprehensive Analysis of Deep Regression. *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**, *42*, 2065–2081. [[CrossRef](#)]
45. Pascual, L.; Gras, M.; Vidal-Brotóns, D.; Alcañiz, M.; Martínez-Máñez, R.; Ros-Lis, J.V. A Voltammetric E-Tongue Tool for the Emulation of the Sensorial Analysis and the Discrimination of Vegetal Milks. *Sens. Actuators B Chem.* **2018**, *270*, 231–238. [[CrossRef](#)]
46. Wójcik, S.; Ciepiela, F.; Jakubowska, M. Computer Vision Analysis of Sample Colors versus Quadruple-Disk Iridium-Platinum Voltammetric e-Tongue for Recognition of Natural Honey Adulteration. *Meas. J. Int. Meas. Confed.* **2023**, *209*, 112514. [[CrossRef](#)]
47. Lee, M. Mathematical Analysis and Performance Evaluation of the GELU Activation Function in Deep Learning. *J. Math.* **2023**, *1*, 4229924. [[CrossRef](#)]
48. Sikandar, S.; Mahum, R.; Als Salman, A.M. A Novel Hybrid Approach for a Content-Based Image Retrieval Using Feature Fusion. *Appl. Sci.* **2023**, *13*, 4581. [[CrossRef](#)]
49. Hasan, M.A.; Haque, F.; Sabuj, S.R.; Sarker, H.; Goni, M.O.F.; Rahman, F.; Rashid, M.M. An End-to-End Lightweight Multi-Scale CNN for the Classification of Lung and Colon Cancer with XAI Integration. *Technologies* **2024**, *12*, 56. [[CrossRef](#)]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.

Article

Pitfalls in Metaheuristics Solving Stoichiometric-Based Optimization Models for Metabolic Networks

Mónica Fabiola Briones-Báez ^{1,*} , Luciano Aguilera-Vázquez ¹ , Nelson Rangel-Valdez ¹ , Cristal Zuñiga ^{2,*} , Ana Lidia Martínez-Salazar ¹  and Claudia Gomez-Santillan ¹ 

¹ División de Estudios de Posgrado e Investigación, Instituto Tecnológico de Ciudad Madero (TECNM), Los Mangos 89440, Mexico; luciano.av@cdmadero.tecnm.mx (L.A.-V.); nelson.rv@cdmadero.tecnm.mx (N.R.-V.); ana.ms@cdmadero.tecnm.mx (A.L.M.-S.); claudia.gs@cdmadero.tecnm.mx (C.G.-S.)

² Department of Biology, San Diego State University, 5500 Campanile Drive, San Diego, CA 92182, USA

* Correspondence: d00070469@cdmadero.tecnm.mx (M.F.B.-B.); czuniga2@sdsu.edu (C.Z.)

Abstract: Flux Balance Analysis (FBA) is a constraint-based method that is commonly used to guide metabolites through restricting pathways that often involve conditions such as anaplerotic cycles like Calvin, reversible or irreversible reactions, and nodes where metabolic pathways branch. The method can identify the best conditions for one course but fails when dealing with the pathways of multiple metabolites of interest. Recent studies on metabolism consider it more natural to optimize several metabolites simultaneously rather than just one; moreover, they point out the use of metaheuristics as an attractive alternative that extends FBA to tackle multiple objectives. However, the literature also warns that the use of such techniques must not be wild. Instead, it must be subject to careful fine-tuning and selection processes to achieve the desired results. This work analyses the impact on the quality of the pathways built using the NSGAI and MOEA/D algorithms and several novel optimization models; it conducts a study on two case studies, the pigment biosynthesis and the node in glutamate metabolism of the microalgae *Chlorella vulgaris*, under three culture conditions (autotrophic, heterotrophic, and mixotrophic) while optimizing for three metabolic intermediaries as independent objective functions simultaneously. The results show varying performances between NSGAI and MOEA/D, demonstrating that the selection of an optimization model can greatly affect predicted phenotypes.

Keywords: cell metabolism; FBA; multi-objective optimization; NSGAI; MOEA/D



Citation: Briones-Baez, M.F.; Aguilera-Vazquez, L.; Rangel-Valdez, N.; Zuñiga, C.; Martinez-Salazar, A.L.; Gomez-Santillan, C. Pitfalls in Metaheuristics Solving Stoichiometric-Based Optimization Models for Metabolic Networks. *Algorithms* **2024**, *17*, 336. <https://doi.org/10.3390/a17080336>

Academic Editors: Massimiliano Caramia and Frank Werner

Received: 30 March 2024

Revised: 1 July 2024

Accepted: 16 July 2024

Published: 1 August 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Microalgae are photosynthetic cellular microorganisms that have been known since the beginning of time. They can be grown either in wastewater or in clean or salty waters; some strains, such as *Dunaliella salina*, can be grown in salty waters, and other strains, such as *Chlorella vulgaris*, are grown in fresh water and can survive high growing temperatures. Microalgae need a carbon source to carry out photosynthesis and produce their biomass. Carbon can be obtained as CO₂ from polluting sources, and they transform it into oxygen, circularly helping to reduce global warming [1].

Microalgae are large producers of biomass; inside there are metabolites such as lipids that in the future can be used as biofuels, amino acids and pigments that are currently used in the pharmaceutical and cosmetic industries, and proteins that are used as food supplements [2,3]. In addition, microalgae present biotechnological applications as bioremediation sources of water quality and have been used as alternatives for the removal of heavy metals due to some strains, such as *Chlorella vulgaris* and *Scenedesmus obliquus*, being able to absorb heavy metals such as Cadmium (Cd) and Lead (Pb) [4,5].

All the characteristics above make the study of the metabolism of microalgae attractive for metabolic engineering. This discipline focuses on the study of the topography of the

network, the regulation of a metabolic pathway, the identification of bottlenecks, the determination of metabolic fluxes, and the elimination of side reactions by gene deletion [6].

In particular, various metabolic engineering techniques have been used to analyze pathways and optimize fluxes to manipulate metabolism and modify the fluxes towards a desired product and thus be able to add commercial value. However, metabolic fluxes cannot be measured *in vivo*; for this reason, modeling approaches are required to measure or predict them [7]. Among them are single-objective constraint-based approaches, such as FBA (Flux Balance Analysis), exact mathematical multi-objective, and heuristic-based approaches.

FBA is one of the most used techniques for studying cellular metabolism is the single-objective FBA approach based on constraints. This approach is widely used in the analysis of the fluxes of metabolic networks since it can be used even if kinetic data are not available, but it requires information on the stoichiometric data of the reactions present in the network, growth requirements, and parameter-specific measurement methods of the biological system, in particular the reconstruction of the metabolic network for the genome scale [8] that include all known reactions that are present in the studied organisms and the genes that encode each enzyme [9].

Cellular metabolism in metabolic modeling is described as the set of chemical reactions present in an organism. This is mathematically represented by a stoichiometric matrix, S , of size $(m \times n)$, where n are the reactions and m are the metabolites involved in each reaction, assigning a negative coefficient if it represents a reactant and a positive coefficient if it is a product, and a coefficient of zero means that the metabolite is not present in the reaction; each reaction will have a lower bound and an upper bound limiting the space of solutions or the maximum and minimum value of the allowed flux. The FBA seeks the linear optimization of an objective function; this function represents the linear combination of the fluxes that generally represents biomass production [10].

$$\begin{aligned} \text{FBA} \quad & \max F(v) = v_{\text{biomass}} \\ \text{Subject to} \quad & S \cdot v = 0 \\ & LB_j \leq v_j \leq UB_j, \forall j \in \{1, \dots, n\} \end{aligned} \quad (1)$$

Equation (1) defines the associated FBA linear optimization problem [11], where v is the flux vector across the reactions. The stoichiometric matrix $S_{m \times n}$ represents the metabolic network, where there exists a metabolite per row and a reaction per column. The value of the cell S_{ij} is the stoichiometric coefficient of the metabolite i involved in reaction j [9], and LB_j and UB_j are the lower and upper bounds for the fluxes allowed in the metabolic system. The steady-state assumption is established by $Sv = 0$ [12].

Its versatility has meant that FBA has been widely used in different organisms, including microalgae, bacteria, consortia of microorganisms, etc. An example of this is the prediction of cell growth of the cyanobacteria *Synechocystis* in [13]; it has also been used in the degradation of glucose by anaerobic digestion to predict the distribution of metabolites and reveal the transformation of carbon in order to evaluate the conversion of ethanol, propionic acid, and butyric acid into acetic acid [14]. FBA has served as a study in medicine, where flux activities were calculated to study differences in metabolic pathways, comparing breast cancer subtypes [15].

The FBA methodology has been used to evaluate metabolic fluxes in different strains of microalgae using three heterotrophic and mixotrophic autotrophic growth conditions. The first microalgae to use FBA was the microalgae *Chlamydomonas reinhardtii* [16], with which the first metabolic map was obtained. Later, the microalga *Chlorella vulgaris* was utilized for the study of lipid production [17]; both had biomass production as their sole objective function.

Although this approach has been widely used, the distribution of metabolites through the pathways is conditioned by the metabolites present in the objective function equation and in the experimental parameters; blocking some metabolic pathways results in

a distribution of fluxes with zero values. Moreover, FBA has been widely used in the search for maximizing the production of compounds of interest, but cellular metabolism in its natural state does not guide metabolic pathways toward the production of a particular metabolite. In the search for a better understanding of microalgae metabolism, new optimization techniques have emerged, such as metaheuristics for multi-objective optimization [18]. These techniques seek a more uniform distribution and are closer to the reality of what happens at the metabolic level by simultaneously optimizing various functions with conflicting objectives.

Multi-objective optimization is generally based on the search for solutions to different conflicting objectives that must be optimized simultaneously. Multi-objective optimization is of great importance and has been carried out at a technological and scientific level. Some examples in the chemical industry have been reported in optimizing operating unit processes, biorefinery, reaction engineering, prevention and control, etc. [19]. It has also been utilized in the biology and medicine sector [20], and in metabolic engineering [18].

Multi-objective optimization contrasts with open-access Cobrapy FBA, which only maximizes or minimizes one objective function and where only one solution is obtained. In multi-objective optimization, a set of solutions is obtained. The solutions obtained are called non-dominated because no other solution in the search space is better than the others when all objectives are considered simultaneously. This set of solutions is known as Pareto optimal solutions [21].

Between the methods that have been used for multi-objective optimization of microorganisms are evolutionary algorithms such as NSGAII [18], MOMO, based on the Bio-objective model, and exact mathematical methods that spend a lot of computational resources [22].

Metaheuristic algorithms originate from the natural evolution of biological groups; they are part of artificial intelligence and are born from natural computing and heuristic methods (partial search algorithms). Compared to mathematical methods that generate a large computational expense, these methods provide sufficiently good solutions to an optimization algorithm with an acceptable computational time and space [23].

Multi-Objective Evolutionary Algorithms (MOEAs) are widely recognized in the scientific community as an approach to solving multi-objective optimization problems. In particular, the NSGAII (multi-objective EA based on non-dominated classification) proposed by [24] has been quite effective when handling two or three objectives [25,26]. The MOEA/D evolutionary algorithm based on decomposition is tested because it has the characteristic that it works correctly when there are more than three objective functions.

Previously, metaheuristic methods had been used to study the metabolism in [18]. The NSGAII algorithm was developed to optimize three objective functions, proteins, carbohydrates, and CO₂, in a metabolic network of the microalga *Chlamydomonas reinhardtii* using the NSGAII algorithm, a coding scheme based on flux balance analysis (FBA). However, the algorithmic solution might be different for every optimization problem, a difference that can increase depending on how well the involved mathematical model explains the phenomena studied. Metabolic networks might not be exempt from such issues, and these works analyze distinct algorithms (NSGAII and MOEA/D) and distinct optimization models for metabolic networks (four multiobjective optimization approaches) from the perspective of the quality of solutions that might be achieved by them, and the convenience of the information provided. The study is carried out on two case studies that involve intricate conditions involving cycles, bifurcations, and reversible and irreversible reactions. The main contributions of this work include the development of three new multi-objective optimization models for metabolic networks, one new algorithmic solution based on decomposition, and an analysis that guides the proper identification of metaheuristics and models to solve the optimization process behind metabolisms.

The remainder of the document is structured as follows. Section 2 describes the metaheuristics used for the purpose of analysis in this research; particularly, it presents the overall definition and constituents. Section 3 details the novel optimization models for the

metabolic networks proposed in this work. Section 4 discusses the original features included in the design of the NSGAII and MOEA/D used to solve the proposed optimization models. Sections 5 and 6 describes the design of the experiments conducted to test the proposed optimization models and their metaheuristic solutions; it contains the definitions of the cases of studies along with the experiments and the results, concluding with a brief discussion of the observed data.

2. Materials and Methods

This work considers NSGAII and MOEA/D algorithms as the considered metaheuristics to tackle the optimization problem in metabolic networks. The general notion of the design of such algorithms is provided in the remainder of this section.

2.1. NSGAII Algorithm

NSGAII is a Multi-Objective Evolutionary Algorithm (MOEA) that utilizes non-dominated sorting and crowding distance to exert selective pressure toward the Pareto front [24]. The metaheuristic evolves an initial population of P_0 parents using common computable genetic operators such as mutation, crossover, and tournament selection that create a new offspring, Q_t , on each generation. To maintain elitism, the current population, P_t , is combined with the offspring Q_t , and a new population P_{t+1} is chosen based on non-domination ranks and crowding distance to diversify and to break ties when necessary.

The three main components of NSGAII are its fast non-dominated sorting approach, the fast, crowded distance estimation procedure, and the simple crowded comparison operator. The general method derived from NSGAII can be depicted as follows:

1. Initialization of population P_0 of size N using a uniform distribution.
2. Create an offspring population Q_t using binary tournament selection based on crowding-comparison operator, cross-over, and mutation performed on the parent population (P_t), where subscript “ t ” denotes the number of generations. The offspring population and its parent population are combined to produce the entire population R_t , $R_t = P_t + Q_t$. The population R_t will be of size $2N$.
3. Perform a fast nondominated sorting approach on the entire population R_t to identify different fronts of objective functions. $F = \text{fast} - \text{nondominated} - \text{sort}(R_t)$, where $F = (F_0, F_1, F_2, \dots)$ will have in F_0 the non-dominated set of solutions of R_t that best approximates the Pareto frontier.
4. Construct a new parent population (P_{t+1}) of size N from the obtained fronts (F_i). This population of size N is now used for selection, cross-over, and mutation to create a new population (Q_{t+i}) of size N .
5. The process must be repeated until the maximum number of iterations is reached.

2.2. MOEA/D

MOEA/D is a strategy based on decomposing the MOP multiobjective optimization problem (as defined in [27]) into a certain number of scalar optimization subproblems that are optimized simultaneously. Each subproblem is optimized using information that comes exclusively from its neighboring subproblems, achieving less computational complexity in each generation. There are several approaches to transforming a multi-objective problem into a scalar number of optimization problems. One of the most popular approaches is the MOEA/D proposed by [28], where the scalar optimization problems can be formulated as follows:

$$g^{te}(y | w_i, z) = \max\{w_i | f_k(x) - z_k | \} , \quad \text{Subject to: } x \in \Omega \quad (2)$$

where $w = (w_1, w_2, \dots, w_N)$ is a vector of weights and $w_i \geq 0$ for all $i = 1, \dots, k$, and the set $z = \{z_1, z_2, \dots, z_k\}$ is the reference point, where $z_i = \max\{f_i(x) | x \in \omega\}$ for $i = 1, \dots, m$.

For each Pareto optimal point x^* , there exists a vector of weights w , where x^* is the optimal solution of 2.

MOEA/D performs for a certain number of generations, and during each generation it also exerts selective pressure toward the Pareto front using genetic operators such as mutation and crossover. The key element in this strategy is that offspring replace parents based on the scalar function and their closeness at each iteration until a final front is delivered by the algorithm.

3. Proposed Approaches

This work proposes three novel optimization models for metabolic networks that extend FBA to a multi-objective optimization problem. The models, called MOFBA₂, MOFBA₃, and MOFBA₄, represent improvements over the MOFBA₁ proposed in [18] and depicted in Equation (3). MOFBA₁ simultaneously optimizes a set of bioproducts $\{v_{b1}, \dots, v_{bm}\}$ instead of just one, keeps within bounds the reaction fluxes, and satisfies the steady state condition, i.e., they ensure that $Sv = 0$ (where the S is the stoichiometric matrix and v is the fluxes vector).

$$\begin{aligned} \text{MOFBA} \quad & \max F(v) = \{v_{b1}, \dots, v_{bm}\} \\ \text{Subject to} \quad & S \cdot v = 0 \\ & LB_j \leq v_j \leq UB_j, \forall j \in \{1, \dots, n\} \end{aligned} \quad (3)$$

MOFBA₁ is the optimization problem resulting from directly implementing the problem defined in Equation (3). It considers as many objective functions as sets of metabolites of interest. Likewise, it considers as many decision variables as reaction fluxes are needed to define the metabolic system. Note that, for an optimization approach, the search space depends on the decision variables, and based on this definition there is one for each possible flux, i.e., a metaheuristic must search proper flux values within the provided bounds of n distinct decision variables.

The conditions described in the previous paragraph characterize a common pitfall in designing solution strategies for optimization problems. The difficulty appears because metaheuristics might require larger running times to locate feasible solutions when the number of decision variables is large. This work considers this situation and proposes three new optimization models for metabolic networks that reduce the search space (i.e., the number of decision variables that a metaheuristic uses in the search). These models are integrated into an appropriate experimental design to demonstrate that, for quality purposes, it matters whose model one chooses to solve certain problems.

While details on the experiments are provided in further sections, the remainder of this section contains an in-depth description of the three novel MOFBAs proposed in this work, with a summary of their relevance and impact at the end.

3.1. MOFBA₂

In MOFBA₂, as in MOFBA₁, the objective functions to be optimized are the sets of metabolites of interest; hence, the number of objectives, m , is the same. On the other hand, MOFBA₂ considers a reduced set of decision variables consisting of only the reaction fluxes $\{v_{b1}, \dots, v_{bm}\}$ associated with the same metabolites of interest present in the objective function, and an additional one, v , that indicates which of the metabolites of interest leads the search. In other words, MOFBA₂ has $m + 1$ decision variables instead of n . Equation (4) formally defines MOFBA₂.

$$\begin{aligned} \text{MOFBA} \quad & \max F(v) = \{v_{b1}, \dots, v_{bm}\} \\ \text{Subject to} \quad & \text{FBA}(v_{bk}, v) \text{ is feasible} \\ & LB_j \leq v_{bj} \leq UB_j, \forall i \in \{1, \dots, m\} \\ & 1 \leq k \leq m \end{aligned} \quad (4)$$

MOFBA₂ describes a bilevel optimization model where the inner model optimizes the leading metabolite of interest, v , using FBA and delimits the bounds of the metabolites of interest to the ones defined in the outer model. The bounds of the remaining reactions are assumed to be known and fixed according to the analyzed metabolic network.

3.2. MOFBA₃

MOFBA₃ proposes a surrogate model to optimize metabolites of interest. The surrogate model searches for improving two well-known indicators: the Hypervolume (HV) and the Generational Distance (GD). These indicators reflect how well a solution converges to the Pareto front. While the HV must be maximized, the GD must be minimized.

MOFBA₃ has $m + 1$ decision variables, the same ones of MOFBA₂, i.e., the reaction fluxes $\{v_{b1}, \dots, v_{bm}\}$ associated with the same metabolites of interest present in the objective function, and leading metabolite flux v . On the other hand, the number of objectives is always 2, no matter how many metabolites of interest are considered. The distinctive characteristic of this model is its surrogation; instead of directly searching for the proper flux values on the metabolites of interest, it uses indicators of performance in the multi-objective context (i.e., the HV and GD indicators). Equation (5) formally defines MOFBA₃.

$$\begin{aligned} \text{MOFBA} \quad & \min F(v) = -1HV(\{v_{b1}, \dots, v_{bm}\}, R), DG(\{v_{b1}, \dots, v_{bm}\}, Z_r) \\ & \text{Subject to} \\ & \text{FBA}(v_{bk}, v) \text{ is feasible} \\ & LB_j \leq v_{bi} \leq UB_j, \forall i \in \{1, \dots, m\} \\ & 1 \leq k \leq m \end{aligned} \quad (5)$$

This optimization problem assumes that there exists a reference point, R , and a reference set, Z_r . Given that the management of FBA was under the COBRApy package, considering the limits on it, the reference point considered for this work is $R = \{1000, 1000, \dots, 1000\}$. Also, given the availability of an FBA implementation due to the same package, the reference set Z_r is formed by the set of optimal solutions formed by those obtained when solving FBA to optimality, having each metabolite of interest as optimized biomass.

3.3. MOFBA₄

MOFBA₄ is the last proposed optimization problem combining the ideas of MOFBA₂ and MOFBA₃. That is, it proposes to optimize not only the metabolites of interest but also the indicators HV and GD. The number of decision variables for this model remains as $m + 1$, and the number of objectives is $m + 3$. Equation (6) formally defines this optimization problem.

$$\begin{aligned} \text{MOFBA} \quad & \min F(v) = -1HV(\{v_{b1}, \dots, v_{bm}\}, R), DG(\{v_{b1}, \dots, v_{bm}\}, Z_r)(v_{b1}, \dots, v_{bm}) \\ & \text{Subject to} \\ & \text{FBA}(v_{bk}, v) \text{ is feasible} \\ & LB_j \leq v_{bi} \leq UB_j, \forall i \in \{1, \dots, m\} \\ & 1 \leq k \leq m \end{aligned} \quad (6)$$

3.4. Analysis

Table 1 summarizes the most notable features of the optimization models proposed in this work, and compares them against the MOFBA₁ proposed in [18]. The search space is greatly reduced in the novel models, and some of them use convergence information in their definition. The unique characteristics demonstrate the richness of the models that can be designed to solve a specific optimization problem.

Table 1. Relevant features of MOFBA models for metabolic networks.

Model	No. Decision Variables	No. Objectives	Surrogate
MOFBA ₁	n	m	No
MOFBA ₂	$m + 1$	m	No
MOFBA ₃	$m + 1$	2	Yes
MOFBA ₄	$m + 1$	$m + 2$	Yes

The proposed MOFBAs cannot be solved with traditional linear solvers such as FBA. The alternatives are to use enumerative schemes or approximate approaches that allow one to obtain solutions belonging to the Pareto optimal frontier. In this sense, this research analyzed the use of metaheuristics that integrate FBA in their search process as an appropriate solution approach since they improve their approximation to the Pareto front in each iteration.

4. Metaheuristic Designs

This section presents the particular details required in this research for the implementation of the NSGAII and MOEA/D metaheuristics to solve the four optimization problems MOFBA₁, MOFBA₂, MOFBA₃, and MOFBA₄. These metaheuristics are based on the NSGAII and MOEA/D frameworks.

The metaheuristics considered require the definition of the following characteristics: (1) coding schemes; (2) fitness evaluation function; (3) genetic operators; and (4) constraint management strategy. The population initialization method for both strategies (NSGAII and MOEA/D) is random. The proposed design for the rest of these components to handle the novel MOFBAs is detailed in the remainder of this section. The novel adaptations for the NSGAII and MOEA/D frameworks include the clever computable representation of solutions associated with the coding schemes.

4.1. Coding Schemes

This work proposes the use of distinctive solution coding sets for each MOFBA (as defined in Equations (3)–(6)). The coding schemes involve the definition of a data structure that represents a solution of the metabolic network. The script developed for experimentation is found in this Github repository (<https://github.com/multiobjectiveoptimization2/MOFBAs>, accessed on 24 June 2024).

For MOFBA₁ the data structure is a real-valued vector, $\mathcal{V}$. The coding scheme considered a metabolic network, $\mathcal{M}_N$, constituted by a set of reactions, $\mathcal{V}$, and two subsets $\mathcal{V}^M, \mathcal{V}^b \subseteq \mathcal{V}$, where $\mathcal{V}^M \cap \mathcal{V}^b = \emptyset$, which represent the reactions of the metabolites of interest to a decision maker. Furthermore, let $v = (v_1, \dots, v_n)$ be the flux vector for $\mathcal{V}$ and assume that there are initial lower and upper bounds, LB_i, UB_i , for each $v_i, 1 \leq i \leq n$. Then, the $\mathcal{W}$ encoding scheme proposes redefining the boundaries of each v_i associated with a reaction in $\mathcal{V}^M \cup \mathcal{V}^b$ using two values (I_i, Δ_i) . The new limits are calculated as $LB_i^{new} = I_i$ and $UB_i^{new} = (UB_i - I_i)\Delta_i + I_i$. All remaining fluxes will keep their limits unchanged. In other words, the solution encodes boundary changes for FBA to solve $\mathcal{M}_N$ using a prespecified bioproduct, which in this work is assumed to be v_1^b . The resulting encoding vector W is of size $O(n)$, asymptotical in the number of reactions.

For MOFBA₂, MOFBA₃, and MOFBA₄, a vector of size $m + 1$ is considered as the encoding scheme, where m is the number of objective functions. The first m elements of the vector are real variables whose value represents the upper limit of flux for each metabolite of interest in the associated reactions. The additional element is a single-objective optimization selector that takes values between 1 and m , indicating which metabolite is going to be optimized in turn. When it is a reversible reaction, the value in the indicated variable will be the same for the lower bound (but negative).

4.2. Fitness Evaluation Function

Between all MOFBA optimization problems, the fitness evaluation (or FEA) functions are considered a derived subset of the set composed of the values of the metabolite fluxes of interest, such as the Hypervolume and Generational Distance metrics. Since the required information on bioproducts is associated with specific reactions, the suitability of a solution obtained by metaheuristics on MOFBAs is evaluated considering their flux values. In MOFBA₁ and MOFBA₂, the criteria or objective functions to be optimized will be the reaction fluxes corresponding to the bioproducts of interest chosen in $\mathcal{V}^b$ and denoted as $(v_1^b, \dots, v_m^b)$. In MOFBA₃, the Hypervolume and Generation Distance obtained from a solution, the reference point R , and the reference set Z_r , defined as follows, are optimized. The R point is the worst possible extreme value of fluxes, which is 1000 for any metabolite of interest, considering its definition for FBA, in widely used platforms, such as CobraPy. The set Z_r is made up of three points, which include the optimal fluxes obtained by solving the case in question using the FBA method by individually optimizing each metabolite of interest; therefore, if there are n objective functions, Z_r will have a cardinality of n . It is worth mentioning that when a leading bioproduct is required in MOFBA₂ to MOFBA₄, this is chosen derived from the value of one of the decision variables considered, as previously described.

4.3. Genetic Operators

These operators create new solutions by dynamically and randomly varying the values of the decision variables in the existing solutions. This selection was due to its success in solving problems involving decision variables with real values [29]. The operators chosen for NSGAII are mutation, crossover, and a simple but reliable random selection, respectively. The specific values of these parameters were taken from the literature and are shown in Table 2.

Table 2. NSGAII implementation-specific parameters.

Parameter	Value
Polynomial Mutation	Probability = $1.0/d$, where d is the number of decision variables. Distribution Index = 20
SBXCrossover	Probability = 100% Distribution Index = 20
Stoppage Criterion	until reaching 100,000 evaluations
Population Size	100

The operators chosen for MOEA/D for mutation and crossover were Polynomial mutation [30], with crossover by differential evolution. The selection strategy is simple but reliable, and the aggregation function used was the Tshebycheff distance. For a more extensive reference of operators, see [28]. The specific values of these parameters were taken from the literature and are shown in Table 3.

Table 3. MOEA/D implementation-specific parameters.

Parameter	Values
Polynomial Mutation	Probability = $1.0/d$, where d is the number of decision variables. Distribution Index = 20
Differential Evolution	CR = 1 F = 0.5 K = 0.5
Stoppage Criterion	Upon completion of 100,000 evaluations
Population Size	100

4.4. Constraint Management Strategy

This work uses the constraint management method proposed in [31] to generate selective pressure towards feasible solutions. As generations evolve in both metaheuristics, the competition between solutions will always prefer the feasible solution despite the non-domination state. In the long run, such a strategy tends to eradicate infeasible solutions in the final algorithm report. Multi-objective optimization is used when there are several objectives to optimize simultaneously. Several multi-objective evolutionary algorithms (MOEAs) exist, such as NSGAII and MOEA/D. Although they are used for optimizing multi-objective problems, they are significantly different. NSGAII is a non-dominated classification algorithm, while the MOEA/D algorithm is based on decomposition.

5. Design of Experiments

This subsection presents the set of experiments performed in order to validate the application of the proposed optimization models as tools to improve the understanding of microalgal metabolisms. In the field of research on effective solutions to multi-objective problems, experiments were conducted on two networks of the microalgae *Chlorella vulgaris* to evaluate the performance of different algorithms and the respective MOFBAs. The subsections present the case of studies, the experimental design, and the software details used in implementation, with the purpose of verifying the following hypotheses.

Hypotheses 0 (H_0). *It is not relevant to the selection of model and/or solution algorithm to optimize fluxes in a metabolic network.*

5.1. Cases of Study

Compared to a previous investigation [18], two networks, glutamate metabolism and pigment flux distribution of the microalgae *Chlorella vulgaris*, were included in the two case studies; reversible and irreversible reactions were added, the representation of a reversible reaction using the intervals of lower bound fluxes of -1000 and upper bound 1000 , and the irreversible ones with intervals of lower bound 0 to upper bound 1000 . In addition, nodes were included where the metabolites bifurcate towards different routes, and cycles that are frequently presented in the metabolism of the cells.

5.1.1. Case of Study 1: Metabolic Network *Chlorella vulgaris*

The metabolic network of the microalga *Chlorella vulgaris* [17] was studied using NSGAII and MOEA/D algorithms for three different culture conditions: photoautotrophy (light + components), heterotrophy (component), and mixotrophy (CO_2 + light + component). Among the compounds that were used as nutrients for cultivation were the addition of nitrogen sources, such as NO_3 and NH_4 , as well as sulfates, such as SO_4 , Fe_2 , and Magnesium. The different crop sources affect the production of metabolites. The following figure shows the distribution of pigments in the microalga *C. vulgaris*.

In this case, a part of the pigment distribution network will be studied since the distribution in microalgae such as *C. vulgaris* is of great importance in the study of pigment synthesis. The reactions involved in the metabolism are presented in Appendix A.1, Table A1. This network includes the complexity of reversible and irreversible (FRDPth, GRDPth) reactions, nodes, and cycles, and are showed in the Figure 1.

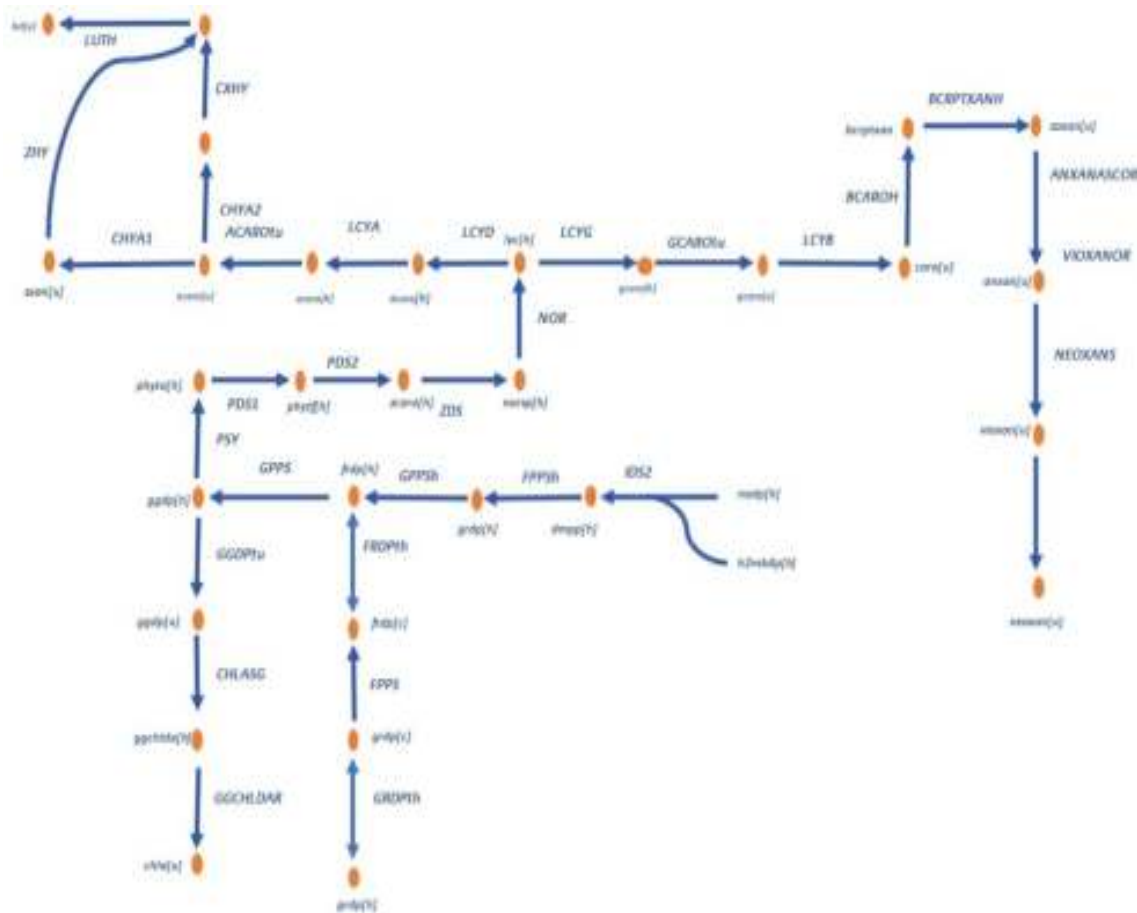


Figure 1. Flux distribution of pigment biosynthesis pathways [17].

5.1.2. Case of Study 2: Optimization Multiobjective of the Metabolic Network of Metabolism Glutamate of Microalgae *Chlorella vulgaris*

Metabolism consists of different metabolic pathways that are intertwined to form a more complex one. One such pathway is the distribution of fluxes of glutamate metabolism, which serves different functions, including amino acid synthesis.

This metabolic network represents great complexity due to the number of metabolites that branch at the central node and the presence of reversible reactions such as ASPATH and ASPNA1Th. This network was evaluated in three different growth conditions, autotrophy, heterotrophy, and mixotrophy, using the NSGAII and MOEA/D algorithms with their four MOFBAs. This metabolism is of great importance because the pathways for producing different products of interest, amino acids such as tyrosine, valine, leucine, etc., are involved, which can later be used to produce proteins. The reactions involved in the metabolism are presented in Appendix A.1, Table A2. Figure 2 represents the distribution of fluxes associated with glutamate metabolism in the chloroplast and cytoplasm.

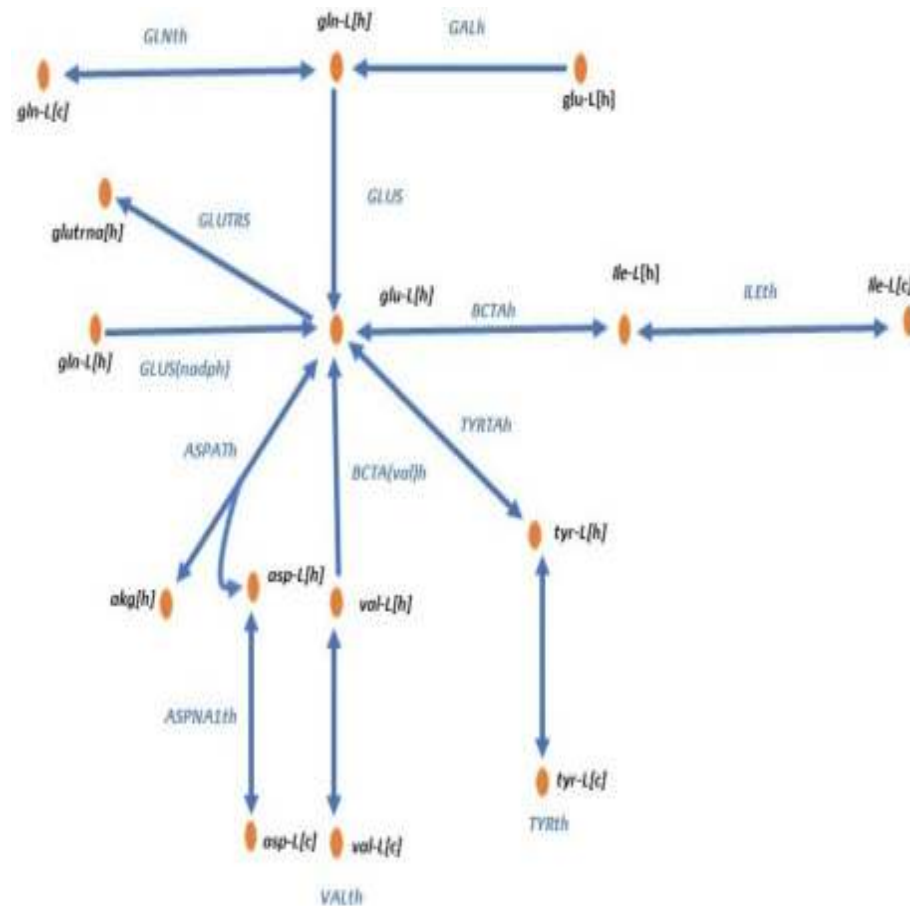


Figure 2. Flux distribution associated with glutamate metabolism in the chloroplast and cytoplasm [17].

5.2. Experiments Definition

Defining a methodology based on metaheuristics that improve the metabolic network flux information provided by the FBA method by redefining Flux Balance Analysis as a Multi-objective Optimization Problem is possible; four experiments were proposed, summarized in the following Table 4.

Table 4. Design of experiments to demonstrate metaheuristic support for understanding metabolism in microalgae.

Experiment	Objective	Variables Involved
Experiment ₁	Using different optimization models produces different results.	<i>C. vulgaris</i> , NSGAIL, MOFA ₁ , MOFA ₂ , MOFA ₃ , MOFA ₄ .
Experiment ₂	Demonstrate that the use of metaheuristics supports the understanding of microalgae metabolism.	<i>C. vulgaris</i> , FBA, NSGAIL, MOFA ₄ .
Experiment ₃	There are optimization algorithms more suitable for solving specific problems	<i>C. vulgaris</i> , NSGAIL, MOEA/D, MOFA ₄ .
Experiment ₄	Validate that a random selection is not enough.	<i>C. vulgaris</i> , NSGAIL, FBA, random.

Experiment 1 demonstrated that different multi-objective optimization models offer different results. The algorithm was set to NSGAI, the microalgae was modified to demonstrate the approach's versatility, and finally, the four proposed optimization models were analyzed. For the cases studied, it was observed that the consistently best model was MOFBA₄, which was used in the subsequent experiments.

In Experiment 2, the algorithm, the microalgae, and the single-objective FBA model were compared against the multi-objective model defined by MOFBA₄. It was observed that the metabolism of a microalgae can be described with different fluxes, not one, and these can be controlled to obtain information on different metabolites of interest. This confirms the ease of adaptation of the proposed methods to different types of metabolic networks, considering different configurations.

Experiment 3 evaluated the performance of the NSGAI and MOEA/D algorithms on the same optimization problem. It was observed that, for the three objectives (i.e., three metabolites of interest) considered, NSGAI was the best. This result is consistent with the literature, given that for two or three objectives, NSGAI shows better performance than MOEA/D. This leaves open the question of whether MOEA/D will improve for a larger number of objectives, which is an open line of investigation for its application in the study of microalgal metabolic networks.

Experiment 4 shows that simple random sampling is not sufficient to obtain a better distribution of solutions, which is possible through the use of metaheuristics.

The metaheuristics NSGAI and MOEA/D were implemented with the aid of the jMetalPy framework [32]. The optimization models were developed in Python and used as part of the FBA implementation provided by the package COBRApy [33]. Graphics were recreated using the interface pyplot of matplotlib [34]. The computer used to run the experiments has a 64 bit 2.6 GHz processor with 32 RAM memory.

6. Results

This section summarizes the data obtained as a result of the implementation of the experiments described in Section 3.2. At the end, it provides a discussion over the achieved goals in the research. To visualize the results in Figures 3–6, the matplotlib library was imported.

6.1. Experiment 1

En [18] showed that NSGAI presents better quality solutions than the classic FBA for three objective functions. Figure 3 shows the results of the experiment where the four variants described above, MOFBA₁, MOFBA₂, MOFBA₃, and MOFBA₄, for the NSGAI algorithm are tested, with the optimization of three objective functions of the pigment flux distribution of the microalgae *Chlorella vulgaris*. It can be observed that the different MOFBAs offer different behaviors to each other; MOFBA₁, MOFBA₃, and MOFBA₄ improve FBA. However, the best solution behavior was MOFBA₄ in Figure 3d as it provides more non-dominated solutions and maintains a good population diversity for the same population size environment.

Figure 3 shows the pigment flux distribution in the microalgae *C. vulgaris*; it can be seen that each variant of MOFBA offers different behaviors and all improve the FBA in Figure 3a–c, but the best behavior in the solutions can be observed in the variant of MOFBA₄ in Figure 3d.

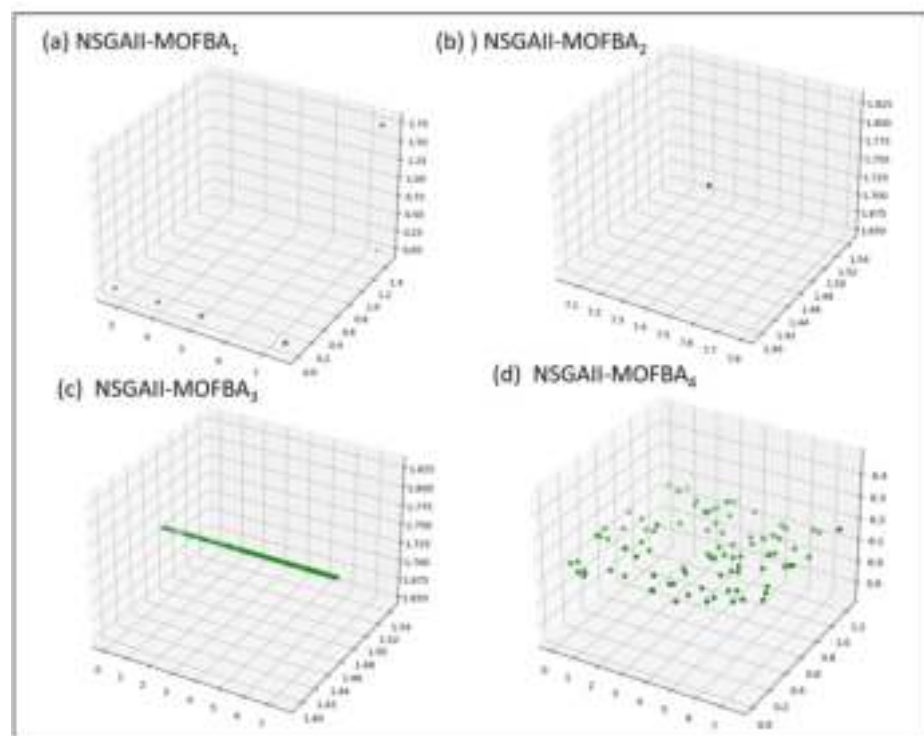


Figure 3. Comparison between the NSGAI algorithm and the variants (a) MOFBA₁, (b) MOFBA₂, (c) MOFBA₃, and (d) MOFBA₄ in the distribution of pigment fluxes.

6.2. Experiment 2

When comparing the performance of the NSGAI algorithm with the MOFBA₄ variant, Figure 4a and the classic single-objective optimization FBA in Figure 4b, it can be observed that the NSGAI-MOFBA₄ algorithm presents superiority by demonstrating that the information provided is improved by providing more solutions and, importantly, a significantly improved distribution. This enhanced distribution is particularly evident in Figures 3d and 4a with two different metabolic networks studied, such as glutamate metabolism and the distribution of pigment fluxes in the microalgae *C. vulgaris*.

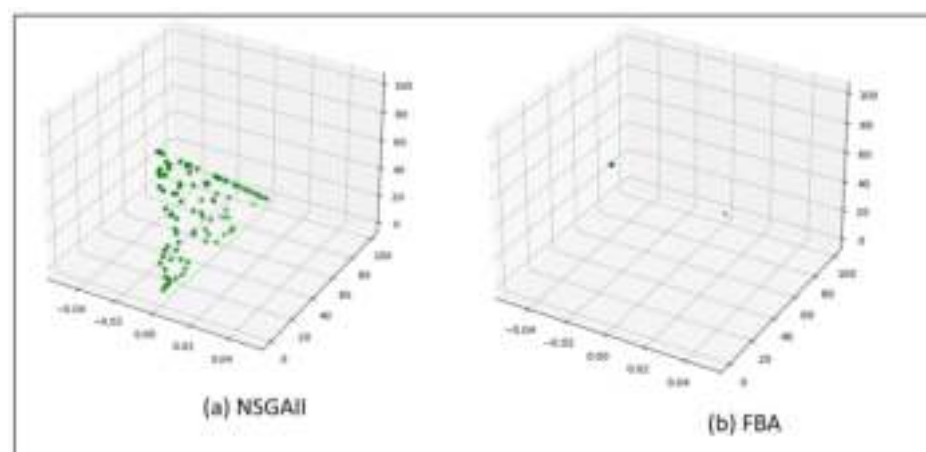


Figure 4. Comparison between (a) NSGAI-MOFBA₄ and (b) FBA in the distribution of fluxes associated with the glutamate metabolism of the microalgae *Chlorella vulgaris*.

6.3. Experiment 3

Figure 5 presents the evaluation between the NSGAI, Figure 5a,c, and MOEA/D, Figure 5b,d, algorithms in the distribution of fluxes associated with glutamate metabolism

and the distribution network and pigment fluxes with NSGAI, Figure 5c, and MOEA/D, Figure 5d, from the microalgae *C. vulgaris* with the variant of the MOFBA₄ algorithm. It was shown that the NSGAI algorithm has more solutions and better population diversity compared to MOEA/D. Metaheuristics are important; in this case, NSGAI is the best, which is consistent with the literature [35], because this algorithm works well with 2 and 3 objectives.

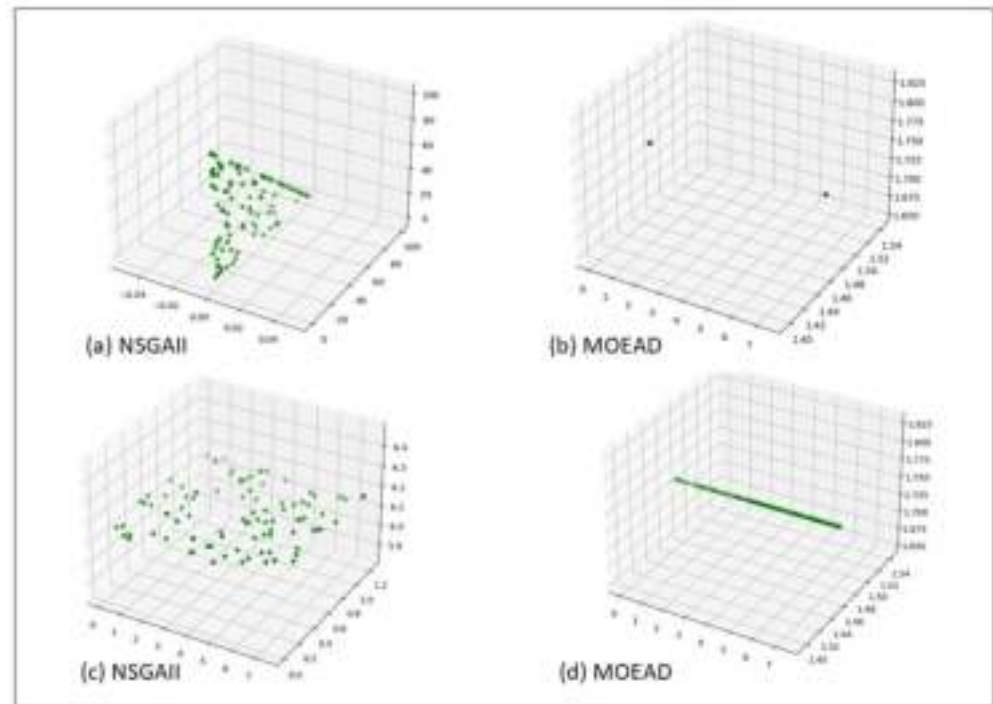


Figure 5. Comparison between the (a,c) NSGAI and (b,d) MOEA/D algorithms in the pigment distribution network and in the distribution of fluxes associated with glutamate metabolism of the microalgae *Chlorella vulgaris*.

6.4. Experiment 4

In addition to testing the different case studies with the FBA, NSGAI, and MOEA/D approaches, an experiment was carried out using a rapid random approach, which we call random, in the distribution of pigment flux in the microalgae *C. vulgaris*. Figure 6 shows the comparison between FBA, Figure 6a, random, Figure 6c, and NSGAI, Figure 6b. The random method, despite being fast, could not offer better results, Figure 6b, compared to what is presented in Figure 6c, NSGAI.

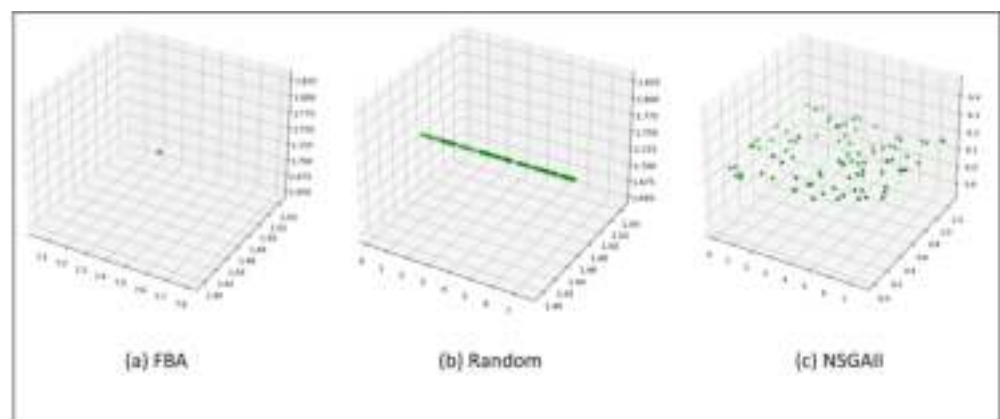


Figure 6. Comparison between the variants (a) FBA, (b) random, and (c) NSGAI in the microalgae *C. vulgaris*.

6.5. Statistic Analysis

Tables 5 and 6 shows that the proposed methods obtain feasible solutions; it is shown that they satisfy the conditions identified in [17] and FBA, thereby demonstrating the correlation of the solutions in silico and its ability to emulate results in different culture conditions. In Table 6 can be seen a comparison between the fluxes in mmol h^{-1} obtained with FBA and NSGAII-MOFBA₄ and the Euclidean distance presented between them. Table 5 demonstrates that NSGAII presents great versatility to limit the parameters in the growing conditions through the lower bound and upper bound values, in addition to being able to simulate cycles and bifurcations between metabolic networks. Likewise, some feasible solutions corresponding to NSGAII with the MOFBA₄ variant and solutions obtained from the classic FBA are presented.

Table 5. Distribution of fluxes in mmol h^{-1} obtained by NSGA II, associated with the synthesis of pigments in the metabolism of the microalgae *C. vulgaris*.

Reaction	LB	UB	S1	S2	S3	S4
NADPH	0.000487	0.000487	0.000487	0.000487	0.000487	0.000487
IDS2	0	0.000487	7.42×10^3	4.72 8	2.58×10^{-5}	4.33×10^{-7}
FPPSh	0	0.000487	7.42×10^{-5}	4.72×10^{-5}	2.58×10^{-5}	4.33×10^{-7}
GPSh	0	0.000487	7.42×10^{-5}	4.72×10^{-5}	2.58×10^{-5}	4.33×10^{-7}
FRDPth	0	0	0	0	0	0
FPPS	0	0	0	0	0	0
GRDPth	0	0	0	0	0	0
GRDPH	0	0	0	0	0	0
GGPPS	0	0.000487	7.42×10^{-5}	4.72×10^{-5}	2.58×10^{-5}	4.33×10^{-7}
v1	0	0.000402078	0	0	0	0
GGDPtu	−0.0000849	0.0000849	7.42×10^{-5}	4.72×10^{-5}	2.58×10^{-5}	4.33×10^{-7}
CHLASG	0	0.0000742	7.42×10^{-5}	4.72×10^{-5}	2.58×10^{-5}	4.33×10^{-7}
GGCHLDAR	0	0.0000742	7.42×10^{-5}	4.72×10^{-5}	2.58×10^{-5}	4.33×10^{-7}
CHLAU	0	0.0000742	7.42×10^{-5}	4.72×10^{-5}	2.58×10^{-5}	4.33×10^{-7}
PSY	0	2.24×10^{-8}	0	0	0	0
PDS1	-2.24×10^{-8}	2.24×10^{-8}	0	0	0	0
PDS2	-2.24×10^{-8}	2.24×10^{-8}	0	0	0	0
ZDS	-2.24×10^{-8}	2.24×10^{-8}	1.59×10^{-8}	1.54×10^{-8}	1.06×10^{-8}	3.58×10^{-9}
NOR	-2.24×10^{-8}	2.24×10^{-8}	1.59×10^{-8}	1.54×10^{-8}	1.06×10^{-8}	3.58×10^{-9}
v2	0.00	8.00×10^{-11}	0	0	0	0
LCYG	-1.59×10^{-8}	1.59×10^{-8}	1.59×10^{-8}	1.54×10^{-8}	1.06×10^{-8}	3.58×10^{-9}
GCAOtu	-1.59×10^{-8}	1.59×10^{-8}	1.59×10^{-8}	1.54×10^{-8}	1.06×10^{-8}	3.58×10^{-9}
LCYB	-1.59×10^{-8}	1.59×10^{-8}	1.59×10^{-8}	1.54×10^{-8}	1.06×10^{-8}	3.58×10^{-9}
v3	0.00	1.24×10^{-8}	1.24×10^{-8}	1.20×10^{-8}	7.10×10^{-9}	1.00×10^{-10}
BCAROH	-3.48×10^{-9}	3.48×10^{-9}	3.48×10^{-9}	3.48×10^{-9}	3.48×10^{-9}	3.48×10^{-9}
BCRPTXANH	-3.48×10^{-9}	3.48×10^{-9}	3.48×10^{-9}	3.48×10^{-9}	3.48×10^{-9}	3.48×10^{-9}
v4	0.00	6.16×10^{-9}	6.16×10^{-9}	6.16×10^{-9}	6.16×10^{-9}	6.16×10^{-9}
ANXANAS	-2.68×10^{-9}	2.68×10^{-9}	-2.68×10^{-9}	-2.68×10^{-9}	-2.68×10^{-9}	-2.68×10^{-9}
v5	0.00	3.88×10^{-9}	3.88×10^{-9}	3.88×10^{-9}	3.88×10^{-9}	3.88×10^{-9}
VIOXANOR	-2.41×10^{-9}	2.41×10^{-9}	-2.41×10^{-9}	-2.41×10^{-9}	-2.41×10^{-9}	-2.41×10^{-9}
v6	0.00	2.70×10^{-10}	2.70×10^{-10}	2.70×10^{-10}	2.70×10^{-10}	2.70×10^{-10}
NEOXANS	-1.47×10^{-9}	1.47×10^{-9}	1.47×10^{-9}	1.47×10^{-9}	1.47×10^{-9}	1.47×10^{-9}
NEOXANU	0	1.47×10^{-9}	1.47×10^{-9}	1.47×10^{-9}	1.47×10^{-9}	1.47×10^{-9}
LCYD	-6.42×10^{-9}	6.42×10^{-9}	0	0	0	0
LCYA	-6.42×10^{-9}	6.42×10^{-9}	0	0	0	0

Table 5. Cont.

Reaction	LB	UB	S1	S2	S3	S4
v7	0.00	1.33×10^{-9}	0	0	0	0
ACAROt _u	-5.09×10^{-9}	5.09×10^{-9}	0	0	0	0
CHYA1	0	0	0	0	0	0
ZHY	0	0	0	0	0	0
CHYA2	0	5.09×10^{-9}	0	0	0	0
CXHY	0	5.09×10^{-9}	0	0	0	0
v8	0	3.35×10^{-9}	-1.74×10^{-9}	-1.74×10^{-9}	-1.74×10^{-9}	-1.74×10^{-9}
LUTH	-1.74×10^{-9}	1.74×10^{-9}	1.74×10^{-9}	1.74×10^{-9}	1.74×10^{-9}	1.74×10^{-9}
LOROXANU	0	1.74×10^{-9}	1.74×10^{-9}	1.74×10^{-9}	1.74×10^{-9}	1.74×10^{-9}
FRDP _{th}	0	1	0	0	0	0

Table 6. Distribution of fluxes in mmol h⁻¹ obtained through FBA and the NSGAII-MOFBA₄ algorithm.

Reaction.	Euclidean Distance between FBA and NSGAII-MOFBA ₄			
	FBA	MOFBA ₄	MOFBA ₄	MOFBA ₄
ACAROt _u '	5.09×10^{-9}	0	0	0
ANXANASCOR'	-2.68×10^{-9}	-2.68×10^{-9}	-2.68×10^{-9}	-2.68×10^{-9}
BCAROH'	3.48×10^{-9}	3.48×10^{-9}	3.48×10^{-9}	3.48×10^{-9}
BCRPTXANH'	3.48×10^{-9}	3.48×10^{-9}	3.48×10^{-9}	3.48×10^{-9}
CHLASG'	0	1.36106×10^{-9}	2.35109×10^{-5}	0.0000742
CHYA1'	0	0	0	0
CHYA2'	5.09×10^{-9}	0	0	0
CXHY'	5.09×10^{-9}	0	0	0
FPPS'	0	0	0	0
FRDP _{th} '	0	0	0	0
GCAROt _u '	1.59×10^{-8}	7.94367×10^{-9}	1.35944×10^{-8}	1.59×10^{-8}
GGCHLDAR'	0	1.36106×10^{-9}	2.35109×10^{-5}	0.0000742
GGDP _{tu} '	0.000148102	1.36106×10^{-9}	2.35109×10^{-5}	0.0000742
GGPS'	0.000486596	1.36106×10^{-9}	2.35109×10^{-5}	0.0000742
GRDP _{th} '	0	0	0	0
IDS1'	0.001459788	1.36106×10^{-9}	2.35109×10^{-5}	0.0000742
LCYA'	6.42×10^{-9}	0	0	0
LCYB'	1.59×10^{-8}	7.94367×10^{-9}	1.35944×10^{-8}	1.59×10^{-8}
LCYD'	6.42×10^{-9}	0	0	0
LCYG'	1.59×10^{-8}	7.94367×10^{-9}	1.35944×10^{-8}	1.59×10^{-8}
LUTH'	1.74×10^{-9}	1.74×10^{-9}	1.74×10^{-9}	1.74×10^{-9}
NEOXANS'	1.47×10^{-9}	1.47×10^{-9}	1.47×10^{-9}	1.47×10^{-9}
NOR'	2.24×10^{-8}	7.94367×10^{-9}	1.35944×10^{-8}	1.59×10^{-8}
PDS1'	2.24×10^{-8}	0	0	0
PDS2'	2.24×10^{-8}	0	0	0
PSY'	2.24×10^{-8}	0	0	0
VIOXANOR'	-2.41×10^{-9}	-2.41×10^{-9}	-2.41×10^{-9}	-2.41×10^{-9}
ZDS'	2.24×10^{-8}	7.94367×10^{-9}	1.35944×10^{-8}	1.59×10^{-8}
ZHY'	0	0	0	0
Euclidean distance		0.001545861	0.001514586	0.001451344

This section statistically validates that there is a difference when using different optimization problems or algorithms in order to show that the choice is relevant. To do this, it summarizes the results by comparing by Hypervolume (the proximity indicator to the Pareto Optimal front) to see whether or not there is a significant difference between the optimization models MOFBA₃ and MOFBA₄ and the NSGAII and MOEA/D algorithms.

The first analysis considers the models MOFBA₃ and MOFBA₄, sets the solution algorithm to NSGAII, and evaluates all networks. For the analysis, the Hypervolume was obtained from each of the 30 runs of the algorithm per problem. Using the non-parametric Wilcoxon signed rank test, with a confidence level of 95%, the H_0 was validated, which specifies that it is impossible to define a methodology based on metaheuristics that improves the information of metabolic network fluxes provided by the FBA method by redefining the Flux Balance Analysis as a Multi-objective Optimization Problem. Table 7 summarizes the results, showing the Hypervolume value on logarithmic scale per run for each network and the acceptance status of the H_0 in the last row. It can be seen that the hypothesis is rejected in almost all the metabolic networks analyzed. Except for *textitChlorella*, it can be commented that the best optimization model is MOFBA₄.

Table 7. Data were statistically analyzed to validate the differences between MOFBA₃ and MOFBA₄. The null hypothesis, H_0 , was accepted when the p -value obtained was less than 0.05.

Glutamate Metabolism in <i>C. vulgaris</i>		Pigment Network in <i>C. vulgaris</i>	
MOFBA ₄	MOFBA ₃	MOFBA ₄	MOFBA ₃
34.539	20.723	20.65	20.647
34.539	20.723	20.65	20.648
34.539	20.723	20.65	20.649
34.539	20.723	20.65	20.649
34.539	20.723	20.65	20.651
34.539	20.723	20.65	20.651
34.539	20.723	20.65	20.651
34.539	20.723	20.65	20.651
34.539	20.723	20.65	20.652
34.539	20.723	20.65	20.648
34.539	20.723	20.65	20.650
34.539	20.723	20.65	20.651
34.539	20.723	20.65	20.648
34.539	20.723	20.65	20.653
34.539	20.723	20.65	20.654
34.539	20.723	20.65	20.652
34.539	20.723	20.65	20.651
34.539	20.723	20.65	20.649
34.539	20.723	20.65	20.651
34.539	20.723	20.65	20.651
34.539	20.723	20.65	20.650
34.539	20.723	20.65	20.650
34.539	20.723	20.65	20.651
34.539	20.723	20.65	20.650
34.539	20.723	20.65	20.650
34.539	20.723	20.65	20.651
34.539	20.723	20.65	20.650
34.539	20.723	20.65	20.650
34.539	20.723	20.65	20.647
34.539	20.723	20.65	20.651
34.539	20.723	20.65	20.651
34.539	20.723	20.65	20.652
34.539	20.723	20.66	20.655
H_0	ACCEPTED	H_0	REJECTED

The second analysis considers the NSGAII and MOEA/D algorithms, sets the optimization model to MOFBA₄, and evaluates all networks. For the analysis, the Hypervolume was obtained from each of the 30 executions of each algorithm on the solved problem. Using the non-parametric Wilcoxon signed rank test, with a confidence level of 95%, the H_0 was validated, which specifies that the difference in means between the samples is the same. Table 8 summarizes the results and shows each network's Hypervolume value per run at the logarithmic scale and the acceptance status of H_0 in the last row. It can be seen that the hypothesis is rejected in all the metabolic networks analyzed. These results from both analyses confirm what was expected, that it is relevant to consider which optimization model to use, and which algorithm, because their performances when obtaining sets of solutions can be different.

Table 8. Data were statistically analyzed to establish differences between the use of NSGAII and MOEA/D. The null hypothesis, H_0 , was accepted when the p -value obtained was less than 0.05.

Glutamate Metabolism in <i>C. vulgaris</i>		Pigment Network in <i>C. vulgaris</i>	
NSGAII	MOEA/D	NSGAII	MOEA/D
20.723	20.723	20.647	20.723
20.723	20.723	20.648	20.723
20.723	20.723	20.649	20.723
20.723	20.723	20.649	20.723
20.723	20.723	20.651	20.723
20.723	20.723	20.651	20.723
20.723	20.723	20.651	20.723
20.723	20.723	20.651	20.723
20.723	20.723	20.652	20.723
20.723	20.723	20.648	20.723
20.723	20.723	20.650	20.723
20.723	20.723	20.651	20.723
20.723	20.723	20.648	20.723
20.723	20.723	20.653	20.723
20.723	20.723	20.654	20.723
20.723	20.723	20.652	20.723
20.723	20.723	20.651	20.723
20.723	20.723	20.649	20.723
20.723	20.723	20.651	20.723
20.723	20.723	20.651	20.723
20.723	20.723	20.650	20.723
20.723	20.723	20.650	20.723
20.723	20.723	20.651	20.723
20.723	20.723	20.650	20.723
20.723	20.723	20.650	20.723
20.723	20.723	20.647	20.723
20.723	20.723	20.651	20.723
20.723	20.723	20.651	20.723
20.723	20.723	20.652	20.723
20.723	20.723	20.655	20.723
H_0	REJECTED	H_0	REJECTED

Through the graphical results observed, mainly due to the volume and dispersion of the solutions obtained in all the algal metabolic networks considered, it is demonstrated that the proposed method based on multi-objective optimization resolved through meta-heuristics offers better support for the analysis. On the other hand, the statistical analysis presented in this section demonstrates that it is relevant to consider the optimization model and the algorithm since these can contribute to different types of improvements.

The statistical analyses presented here demonstrate that there can be a significant difference between optimization models and between metaheuristic algorithms.

6.6. Discussion

Almost no experiment has been done previously with the metabolism of microalgae, except for [18]. Although there are exact methodologies, evolutionary approaches require fewer computational resources in the field of multiple objectives; for example, it gives the advantage of using less time and memory. Approaches such as NSGAI and MOEA/D allow greater power of choice in the decision-making process due to the variety and number of solutions and the possibility of easier recognition of the most important fluxes in a network and their influence and impact, instead of not having a methodology.

Some additional insights emerge from the above results. Experiment 2 demonstrates the versatility of NSGAI to adapt to different circumstances and its ability to improve the analysis of the metabolic network given the greater number of solutions it produces for each of them. As demonstrated in Experiments 1 to 4, the analysis capacity of a metabolic network is improved by introducing the NSGAI algorithm.

The multi-objective optimization problems present in the literature currently consider different solution metrics. Experiment 1 compares the use of the NSGAI algorithm with the four variants of optimization problems, with the MOFBA₄ optimization problem being the most promising, by introducing different optimization strategies, such as optimizing not only the metabolites of interest but also Hypervolume and Generational Distance. Compared to MOFBA₃, which minimizes Hypervolume, in MOFBA₂ the decision variables are only the fluxes of the reactions of interest.

Although, in the case studies, NSGAI had a better graphically observable performance than MOEA/D, as occurred in Experiment 3, because the case studies had three objective functions and, according to the literature, NSGAI is better than MOEA/D when there are three functions objective, the possibility opens up of being able to use MOEA/D in networks where more than three objective functions need to be optimized. It can also be observed, through Experiment 4, that simply using a random sample solution is not enough to obtain a good set of solutions like using metaheuristics. However, special considerations must be taken to allow respect for restrictions or information of control desired by an interested individual.

The algorithms were tested on different microalgae strains, as seen in [18] with *C. reinhardtii*, and in this research using *C. vulgaris*, in complex metabolic networks that contain cycles, bifurcations, and reversible reactions, it checks their viability in different metabolic networks, confirming that they can not only be used in a single microalgae. This leaves open the possibility of it being used in other types of species where there is a need to optimize more than one objective function.

7. Conclusions

The present research work carried out a study of metabolic fluxes in green microalgae. The objective was focused on verifying the suitability of in silico methods as support strategies for improving the analysis of metabolism in microalgae. Through the experiments developed, evidence was obtained that supports the following conclusions:

The study of metabolic fluxes in microalgae is improved through increasing the number of solutions that satisfy the conditions of a microalgae so that it can live. This is observable because, unlike traditional methods such as FBA that only offer a solution, which is expanded in a limited way through sensitivity analysis, it is greatly favored by integrating it into a methodology based on metaheuristics and multi-objective optimization problems; it increases both the number of fluxes that satisfy the conditions sought in the metabolic network, also simultaneously allowing the optimization of several metabolites of interest.

There is more than one alternative to analyzing a metabolic network by optimizing several metabolites of interest, with the FBA method as the core of the optimization process.

The present work proposed four optimization models demonstrating this result, each offering analysis angles different from those that FBA offers.

It is possible to solve the optimization problems supporting the metabolic study by considering different evolutionary metaheuristics, and by obtaining significant results for the analysis. This is demonstrated using NSGAII and MOEA/D to solve the proposed optimization problems. In this study, NSGAII showed the best performance in general, which is consistent with the literature, by exclusively addressing the simultaneous optimization of three objectives. This shows that, for future work, the analysis of the best metaheuristic must be carried out before the study.

Solution search parameters can be controlled during the analysis of a metabolic network by adjusting the reaction boundaries. This contributes to further improving the study of microalgae since the definition of controlled environments is possible. This is observed in the validation process, where the parameters to generate solutions were limited to the values found in the work on in vivo specimens.

Selecting one algorithm or model to optimize a specific metabolic network can be troublesome and requires fine-tuning to identify the configuration that best fits the research interests of the metabolic engineering carried out. This is evident given the variation in the performance between algorithms and optimization models, or the different combinations tested in this research work.

A decision maker, e.g., a researcher in metabolic engineering, improves his decision-making capacity by visualizing a set of metabolic fluxes that satisfy the conditions specified for the metabolic network they study.

Author Contributions: Conceptualization, M.F.B.-B. and L.A.-V.; methodology, M.F.B.-B. and N.R.-V.; software, M.F.B.-B. and N.R.-V.; validation, L.A.-V., A.L.M.-S. and C.Z.; formal analysis, M.F.B.-B., L.A.-V. and N.R.-V.; investigation, M.F.B.-B.; resources, M.F.B.-B. and L.A.-V.; data curation, L.A.-V., A.L.M.-S. and C.Z.; writing—original draft preparation, M.F.B.-B., N.R.-V. and C.G.-S.; writing—review and editing, L.A.-V., A.L.M.-S. and C.Z.; visualization, M.F.B.-B.; supervision, L.A.-V. and A.L.M.-S.; project administration, M.F.B.-B. All authors have read and agreed to the published version of the manuscript.

Funding: C.Z. was supported by the National Science Foundation (Grant No. 2313313). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Data Availability Statement: The script of the modified metaheuristic algorithms NSGAII and MOEA/D with their respective MOFBAs have been stored in the GitHub repository: <https://github.com/multiobjectiveoptimization2/MOFBAs> (accessed on 24 June 2024).

Acknowledgments: The authors acknowledge the support from CONAHCYT projects no. 3058, A1-S-11012, and the Laboratorio Nacional de Tecnologías de la Información (LANTI). They also acknowledge support from the ITCM and from TECNM project 14612.22-P. Mónica Fabiola Briones Báez acknowledges the scholarship No. 784358 from CONAHCYT to pursue his postgraduate studies.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A

Appendix A.1

As previously mentioned, a metabolic network is the set of reactions present in the microorganism. The following list of reactions represents the metabolic network for the synthesis of pigments in the *Chlorella vulgaris* microalgae.

Table A1. Reactions derived from the synthesis of pigments in the metabolism of the microalgae *C. vulgaris* [17].

Name	Formula
FRDPth	$\text{frdp[h]} \rightleftharpoons \text{frdp[c]}$
GRDPth	$\text{grdp[h]} \rightleftharpoons \text{grdp[c]}$
ACAROt	$\text{acar[o]} \rightarrow \text{acar[u]}$
GCAROt	$\text{gcar[o]} \rightarrow \text{gcar[u]}$
GGDPtu	$\text{ggdp[h]} \rightarrow \text{ggdp[u]}$
FPPS	$\text{grdp[c]} + \text{ipdp[c]} \rightarrow \text{frdp[c]} + \text{ppi[c]}$
FPPSh	$\text{grdp[h]} + \text{ipdp[h]} \rightarrow \text{frdp[h]} + \text{h[h]} + \text{ppi[h]}$
GGPS	$\text{frdp[h]} + \text{ipdp[h]} \rightarrow \text{ggdp[h]} + \text{h[h]} + \text{ppi[h]}$
GPPSh	$\text{dmpp[h]} + \text{ipdp[h]} \rightarrow \text{grdp[h]} + \text{h[h]} + \text{ppi[h]}$
IDS2	$\text{h[h]} + \text{h2mb4p[h]} + \text{nadph[h]} \rightarrow \text{dmpp[h]} + \text{h2o[h]} + \text{nadp[h]}$
ANXANASCOR	$\text{anxan[u]} + \text{ascb-L[u]} \rightarrow \text{dhdascb[u]} + \text{h2o[u]} + \text{zaxan[u]}$
BCAROH	$\text{caro[u]} + \text{h[u]} + \text{nadph[u]} + \text{o2[u]} \rightarrow \text{bcrtptxan[u]} + \text{h2o[u]} + \text{nadp[u]}$
BCRPTXANH	$\text{bcrtptxan[u]} + \text{h[u]} + \text{nadph[u]} + \text{o2[u]} \rightarrow \text{h2o[u]} + \text{nadp[u]} + \text{zaxan[u]}$
CHYA1	$\text{acar[u]} + \text{h[u]} + \text{nadph[u]} + \text{o2[u]} \rightarrow \text{h2o[u]} + \text{nadp[u]} + \text{zaxan[u]}$
CHYA2	$\text{acar[u]} + \text{h[u]} + \text{nadph[u]} + \text{o2[u]} \rightarrow \text{crpxan[u]} + \text{h2o[u]} + \text{nadp[u]}$
CXHY	$\text{crpxan[u]} + \text{h[u]} + \text{nadph[u]} + \text{o2[u]} \rightarrow \text{h2o[u]} + \text{lut[u]} + \text{nadp[u]}$
LCYB	$\text{gcaro[u]} \rightarrow \text{caro[u]}$
LCYA	$\text{lyc[h]} \rightarrow \text{dcaro[h]}$
LCYG	$\text{lyc[h]} \rightarrow \text{gcaro[h]}$
NEOXANS	$\text{vioxan[u]} \rightarrow \text{neoxan[u]}$
NOR	$\text{norsp[h]} + \text{o2[h]} + \text{pqh2[h]} \rightarrow 2 \text{h2o[h]} + \text{lyc[h]} + \text{pq[h]}$
PDS1	$\text{phyto[h]} + \text{pq[h]} \rightarrow \text{phytl[h]} + \text{pqh2[h]}$
PDS2	$\text{phytl[h]} + \text{pq[h]} \rightarrow \text{pqh2[h]} + \text{zcaro[h]}$
PSY	$2 \text{ggdp[h]} \rightarrow 2 \text{h[h]} + \text{phyto[h]} + 2 \text{ppi[h]}$
VIOXANOR	$\text{ascb-L[u]} + \text{vioxan[u]} \rightarrow \text{anxan[u]} + \text{dhdascb[u]} + \text{h2o[u]}$
ZDS	$\text{o2[h]} + \text{pqh2[h]} + \text{zcaro[h]} \rightarrow 2 \text{h2o[h]} + \text{norsp[h]} + \text{pq[h]}$
ZHY	$\text{h[u]} + \text{nadph[u]} + \text{o2[u]} + \text{zaxan[u]} \rightarrow \text{h2o[u]} + \text{lut[u]} + \text{nadp[u]}$
CHLASG	$\text{chl[a]} + \text{ggdp[u]} \rightarrow \text{ggchld[a]} + \text{h[u]} + \text{ppi[u]}$
GGCHLDAR	$\text{ggchld[a]} + 3 \text{h[u]} + 3 \text{nadph[u]} \rightarrow \text{chla[u]} + 3 \text{nadp[u]}$
GGDR	$\text{ggdp[h]} + 3 \text{h[h]} + 3 \text{nadph[h]} \rightarrow 3 \text{nadp[h]} + \text{pdp[h]}$
CHLBSG	$\text{chldb[u]} + \text{ggdp[u]} \rightarrow \text{ggchldb[u]} + \text{h[u]} + \text{ppi[u]}$

Table A2. Reactions derived from the flux distribution associated to glutamate metabolism [17].

Name	Formula
GLNth	$\text{gln-L[c]} + \text{h[c]} \rightleftharpoons \text{gln-L[h]} + \text{h[h]}$
GALh	$\text{atp[h]} + \text{glu-L[h]} + \text{nh4[h]} \rightarrow \text{adp[h]} + \text{gln-L[h]} + \text{h[h]} + \text{pi[h]}$
GLUS	$\text{glu-L[h]} + \text{h2o[h]} + \text{nad[h]} \rightarrow \text{akg[h]} + \text{h[h]} + \text{nadh[h]} + \text{nh4[h]}$
GLUTRS	$\text{atp[h]} + \text{glu-L[h]} + \text{trmaglu[h]} \rightarrow \text{amp[h]} + \text{glutma[h]} + \text{h[h]} + \text{ppi[h]}$
GLUS(nadph)	$\text{akg[h]} + \text{gln-L[h]} + \text{h[h]} + \text{nadph[h]} \rightarrow 2 \text{glu-L[h]} + \text{nadp[h]}$
ASPAth	$\text{akg[h]} + \text{asp-L[h]} \rightleftharpoons \text{glu-L[h]} + \text{oa[h]}$
ASPNA1th	$\text{asp-L[c]} + \text{na1[c]} \rightleftharpoons \text{asp-L[h]} + \text{na1[h]}$
VALth	$\text{h[c]} + \text{val-L[c]} \rightleftharpoons \text{h[h]} + \text{val-L[h]}$
BCTA(val)h	$\text{akg[h]} + \text{val-L[h]} \rightarrow 3 \text{mob[h]} + \text{glu-L[h]}$
TYRTAh	$34 \text{hpp[h]} + \text{glu-L[h]} \rightleftharpoons \text{akg[h]} + \text{tyr-L[h]}$
TYRth	$\text{h[c]} + \text{tyr-L[c]} \rightleftharpoons \text{h[h]} + \text{tyr-L[h]}$
BCTAh	$3 \text{mob[h]} + \text{glu-L[h]} \rightleftharpoons \text{akg[h]} + \text{ile-L[h]}$
ILEth	$\text{h[c]} + \text{ile-L[c]} \rightleftharpoons \text{h[h]} + \text{ile-L[h]}$

References

1. Esteves, A.F.; Soares, O.S.; Vilar, V.J.; Pires, J.C.; Gonçalves, A.L. The effect of light wavelength on CO₂ capture, biomass production and nutrient uptake by green microalgae: A step forward on process integration and optimisation. *Energies* **2020**, *13*, 333. [\[CrossRef\]](#)
2. Wang, Y.; Tibbetts, S.M.; McGinn, P.J. Microalgae as sources of high-quality protein for human food and protein supplements. *Foods* **2021**, *10*, 3002. [\[CrossRef\]](#) [\[PubMed\]](#)
3. Brown, M.R.; Jeffrey, S. Biochemical composition of microalgae from the green algal classes Chlorophyceae and Prasinophyceae. 1. Amino acids, sugars and pigments. *J. Exp. Mar. Biol. Ecol.* **1992**, *161*, 91–113. [\[CrossRef\]](#)
4. Kholssi, R.; Ramos, P.V.; Marks, E.A.; Montero, O.; Rad, C. 2Biotechnological uses of microalgae: A review on the state of the art and challenges for the circular economy. *Biocatal. Agric. Biotechnol.* **2021**, *36*, 102114. [\[CrossRef\]](#)
5. Romero, D.C.V.; Cardozo, A.P.; Montes, V.D. Utilización de microalgas como alternativa para la remoción de metales pesados. *RIAA* **2022**, *13*, 10.
6. Woolston, B.M.; Edgar, S.; Stephanopoulos, G. Metabolic engineering: Past and future. *Annu. Rev. Chem. Biomol. Eng.* **2013**, *4*, 259–288. [\[CrossRef\]](#) [\[PubMed\]](#)
7. Kaste, J.A.; Shachar-Hill, Y. Model validation and selection in metabolic flux analysis and flux balance analysis. *Biotechnol. Prog.* **2024**, *40*, e3413. [\[CrossRef\]](#) [\[PubMed\]](#)
8. Anand, S.; Mukherjee, K.; Padmanabhan, P. An insight to flux-balance analysis for biochemical networks. *Biotechnol. Genet. Eng. Rev.* **2020**, *36*, 32–55. [\[CrossRef\]](#) [\[PubMed\]](#)
9. Orth, J.D.; Thiele, I.; Palsson, B.Ø. What is flux balance analysis? *Nat. Biotechnol.* **2010**, *28*, 245–248. [\[CrossRef\]](#)
10. Liu, Y.; Westerhoff, H.V. Competitive, multi-objective, and compartmented Flux Balance Analysis for addressing tissue-specific inborn errors of metabolism. *J. Inher. Metab. Dis.* **2023**, *46*, 573–585. [\[CrossRef\]](#)

11. Raman, K.; Chandra, N. Flux balance analysis of biological systems: Applications and challenges. *Brief. Bioinform.* **2009**, *10*, 435–449. [\[CrossRef\]](#)
12. Lewis, N.; Nagarajan, H.; Palsson, B. Constraining the metabolic genotype-phenotype relationship using a phylogeny of in silico methods. *Nat. Rev. Microbiol.* **2012**, *10*, 291–305. [\[CrossRef\]](#)
13. Toyoshima, M.; Toya, Y.; Shimizu, H. Flux balance analysis of cyanobacteria reveals selective use of photosynthetic electron transport components under different spectral light conditions. *Photosynth. Res.* **2020**, *143*, 31–43. [\[CrossRef\]](#) [\[PubMed\]](#)
14. Huang, J.; Hou, J.; Li, L.; Wang, Y. Flux balance analysis of glucose degradation by anaerobic digestion in negative pressure. *Int. J. Hydrogen Energy* **2020**, *45*, 26822–26830. [\[CrossRef\]](#)
15. Trilla-Fuertes, L.; Gámez-Pozo, A.; Díaz-Almirón, M.; Prado-Vázquez, G.; Zapater-Moros, A.; López-Vacas, R.; Nanni, P.; Zamora, P.; Espinosa, E.; Fresno Vara, J.A. Computational metabolism modeling predicts risk of distant relapse-free survival in breast cancer patients. *Future Oncol.* **2019**, *15*, 3483–3490. [\[CrossRef\]](#)
16. Boyle, N.R.; Morgan, J.A. Flux balance analysis of primary metabolism in *Chlamydomonas reinhardtii*. *BMC Syst. Biol.* **2009**, *3*, 4. [\[CrossRef\]](#) [\[PubMed\]](#)
17. Zuñiga, C.; Li, C.T.; Huelsman, T.; Levering, J.; Zielinski, D.C.; McConnell, B.O.; Long, C.P.; Knoshaug, E.P.; Guarnieri, M.T.; Antoniewicz, M.R.; et al. Genome-scale metabolic model for the green alga *Chlorella vulgaris* UTEX 395 accurately predicts phenotypes under autotrophic, heterotrophic, and mixotrophic growth conditions. *Plant Physiol.* **2016**, *172*, 589–602. [\[CrossRef\]](#) [\[PubMed\]](#)
18. Briones-Baez, M.F.; Aguilera-Vazquez, L.; Rangel-Valdez, N.; Martinez-Salazar, A.L.; Zuñiga, C. Multi-Objective Optimization of Microalgae Metabolism: An Evolutionary Algorithm Based on FBA. *Metabolites* **2022**, *12*, 603. [\[CrossRef\]](#) [\[PubMed\]](#)
19. Rangaiah, G.P.; Petriciolet, A. *Multi-Objective Optimization in Chemical Engineering: Developments and Applications*; Rangaiah, G.P., Bonilla-Petriciolet, A., Eds.; Wiley: New York, NY, USA, 2013.
20. Liu, X.; Tian, J.; Duan, P.; Yu, Q.; Wang, G.; Wang, Y. GrMoNAS: A granularity-based multi-objective NAS framework for efficient medical diagnosis. *Comput. Biol. Med.* **2024**, *171*, 108118. [\[CrossRef\]](#)
21. PDE: A Pareto-frontier differential evolution approach for multi-objective optimization problems. In Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No. 01TH8546), Seoul, Republic of Korea, 27–30 May 2001; Volume 2, pp. 971–978.
22. Andrade, R.; Doostmohammadi, M.; Santos, J.; Sagot, M.F.; Mira, N.P.; Vinga, S. MOMO—multi-objective metabolic mixed integer optimization: Application to yeast strain engineering. *BMC Inform.* **2020**, *21*, 69. [\[CrossRef\]](#)
23. Wang, G.G.; Zhao, X.; Li, K. *Metaheuristic Algorithms: Theory and Practice*; CRC Press: Boca Raton, FL, USA, 2024.
24. Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **2002**, *6*, 182–197. [\[CrossRef\]](#)
25. Cruz, L.; Fernandez, E.; Gomez, C.; Rivera, G.; Perez, F. Many-Objective Portfolio Optimization of Interdependent Projects with ‘a priori’ Incorporation of Decision-Maker Preferences. *Appl. Math. Inf. Sci.* **2014**, *8*, 1517–1531. [\[CrossRef\]](#)
26. Rivera, G.; Florencia, R.; Guerrero, M.; Porras, R.; Sanchez-Solis, J. Online multi-criteria portfolio analysis through compromise programming models built on the underlying principles of fuzzy outranking. *Inf. Sci.* **2021**, *580*, 734–755. [\[CrossRef\]](#)
27. Chang, K.H. Multiobjective optimization and advanced topics. In *Design Theory and Methods Using CAD/CAE*; Elsevier: Amsterdam, The Netherlands, 2015; pp. 325–406.
28. Zhang, Q.; Li, H. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Trans. Evol. Comput.* **2007**, *11*, 712–731. [\[CrossRef\]](#)
29. Cruz-Reyes, L.; Fernandez, E.; Rangel-Valdez, N. A metaheuristic optimization-based indirect elicitation of preference parameters for solving many-objective problems. *Int. J. Comput. Intell. Syst.* **2017**, *10*, 56–77. [\[CrossRef\]](#)
30. Deb, K.; Goyal, M. A combined genetic adaptive search (GeneAS) for engineering design. *Comput. Sci. Inform.* **1996**, *26*, 30–45.
31. Deb, K. An efficient constraint handling method for genetic algorithms. *Comput. Methods Appl. Mech. Eng.* **2000**, *186*, 311–338. [\[CrossRef\]](#)
32. Benítez-Hidalgo, A.; Nebro, A.J.; García-Nieto, J.; Oregi, I.; Del Ser, J. jMetalPy: A Python framework for multi-objective optimization with metaheuristics. *Swarm Evol. Comput.* **2017**, *12*, e0171744. [\[CrossRef\]](#)
33. Ebrahim, A.; Lerman, J.A.; Palsson, B.O.; Hyduke, D.R. COBRAPy: CONstraints-Based Reconstruction and Analysis for Python. *BMC Syst. Biol.* **2013**, *7*, 74. [\[CrossRef\]](#)
34. Hunter, J.D. Matplotlib: A 2D graphics environment. *Comput. Sci. Eng.* **2007**, *9*, 90–95. [\[CrossRef\]](#)
35. Liu, D.; Huang, Q.; Yang, Y.; Liu, D.; Wei, X. Bi-objective algorithm based on NSGA-II framework to optimize reservoirs operation. *J. Hydrol.* **2020**, *585*, 124830. [\[CrossRef\]](#)

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.

Article

Hyperspectral Python: HypPy

Wim Bakker , Frank van Ruitenbeek , Harald van der Werff , Christoph Hecker , Arjan Dijkstra 
and Freek van der Meer 

Faculty ITC, University of Twente, 7522 NH Enschede, The Netherlands; f.j.a.vanruitenbeek@utwente.nl (F.v.R.); c.a.hecker@utwente.nl (C.H.); a.h.dijkstra@utwente.nl (A.D.); f.d.vandermeer@utwente.nl (F.v.d.M.)

* Correspondence: w.h.bakker@utwente.nl (W.B.); h.m.a.vanderwerff@utwente.nl (H.v.d.W.)

Abstract: This paper describes the design, implementation, and usage of a Python package called Hyperspectral Python (HypPy). Proprietary software for processing hyperspectral images is expensive, and tools developed using these packages cannot be freely distributed. The idea of HypPy is to be able to process hyperspectral images using free and open-source software. HypPy was developed using Python and relies on the array-processing capabilities of packages like NumPy and SciPy. HypPy was designed with practical imaging spectrometry in mind and has implemented a number of novel ideas. To name a few of these ideas, HypPy has BandMath and SpectralMath tools for processing images and spectra using Python statements, can process spectral libraries as if they were images, and can address bands by wavelength rather than band number. We expect HypPy to be beneficial for research, education, and projects using hyperspectral data because it is flexible and versatile.

Keywords: hyperspectral imaging; image processing; Python; minerals



Citation: Bakker, W.; van Ruitenbeek, F.; van der Werff, H.; Hecker, C.; Dijkstra, A.; van der Meer, F. Hyperspectral Python: HypPy. *Algorithms* **2024**, *17*, 337. <https://doi.org/10.3390/a17080337>

Academic Editors: Laura Antonelli and Lucia Maddalena

Received: 27 June 2024

Revised: 22 July 2024

Accepted: 30 July 2024

Published: 1 August 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Image processing packages have been around since the creation of the first computers. However, at the start of this century, not many packages were capable of handling hyperspectral image data. Hyperspectral images tend to be huge, and standard operations such as image classification are unfit for handling the high dimensionality of hyperspectral data. Traditionally, hyperspectral images are processed using proprietary packages such as ENVI (ENVI and IDL are trademarks of NV5 Geospatial Solutions, Inc.) (Environment for Visualizing Images) [1] and others. However, with the advent of free open-source software [2], the need was felt to be able to use and process hyperspectral images independently of proprietary packages.

With packages like numpy [3], scipy [4], and matplotlib [5], the Python language [6] has become a data-processing and prototyping environment, rivaling systems (proprietary and free) such as MATLAB (MatLab is a registered trademark of The MathWorks, Inc.), IDL (ENVI and IDL are trademarks of NV5 Geospatial Solutions, Inc.), Octave (Octave is a trademark of Octave Technologies, Inc.), and R.

Python, in addition the aforementioned packages numpy and scipy, is free and open-source. Users can access the source code, make modifications, and distribute their own versions. In principle, any code written in Python can be distributed freely.

We use Python for the implementation of the package Hyperspectral Python, HypPy (pronounced [hip-ee], like the subculture in the 1960s), which is a software package for reading and writing, visualizing, processing, and analyzing hyperspectral data.

HypPy started as a number of image classes and factory functions enabling the reading and writing of ENVI format images. The creation of more tools in HypPy gained momentum when our research group became involved in the Mars Planetary Mapping Pilot Project [7] in 2009 and 2010. Subsequently, various general tools and specific tools for handling Mars hyperspectral data were written. At the same time, HypPy offered a

platform for implementing experimental ideas. Gradually, the package was extended to what it is today. This paper describes the design considerations and implementation of HypPy providing a number of examples.

2. Design and Implementation

In this section we aim to show that HypPy has a set of features, which sets it apart from other packages and makes it a versatile and flexible tool for processing hyperspectral data. HypPy has the following features:

- HypPy supports three levels of interfacing: a graphical user interface (GUI), a command-line interface (CLI), and a programming interface (API). The user interfaces of HypPy can be accessed from one top-level menu.
- The top-level menu is configurable and can be adapted for different purposes.
- HypPy has viewers for hyperspectral images and spectral libraries. These viewers have simple interfaces yet offer sufficient functionality for exploring the quality and information content of the data.
- HypPy offers a read and write capability for ENVI data files.
- Hyperspectral images tend to be huge, yet HypPy is memory-friendly.
- HypPy supports ENVI's three native image formats: BSQ, BIL, and BIP. Internally, HypPy translates these to a BIP format, relieving the programmer of the hassle of having to deal with three different formats.
- HypPy offers functionality for working with wavelengths instead of band numbers. Band numbers have no physical meaning, whereas wavelengths do.
- Raw image data may not have bands in the order of increasing wavelengths. Therefore, HypPy is able to sort wavelengths in the image on-the-fly.
- HypPy is able to use the bad band list (bbl) and skip bands that are marked as bad.
- HypPy can process ENVI spectral libraries as if they were ENVI images.
- If necessary, spectra are resampled on-the-fly. This eliminates the need to spectrally resample before combining spectra from different sources.
- HypPy is free and open-source software, which makes it an ideal tool for education. Students can pick up the package for free and use it at home as well.

In the following sections, the points mentioned above will be discussed in more detail.

2.1. The Usage Levels: GUI, CLI, and API

Image processing packages support different types of interfaces to the software. For instance, the Khoros package [8] support a GUI, CLI, and API. On top of that, Khoros offers an interface for visual programming, which is called Cantata [9]. Building a visual programming environment is complex and is not within the scope of HypPy. However, similar to Khoros, HypPy offers a GUI, CLI, and API.

The advantage of having these three levels is that it offers flexibility. GUIs make it easy to use the tools, even for beginners. From the interface, it is clear which inputs and outputs a certain tool may need. In addition to that, a GUI can be used to invoke multiple tasks at once. In this way, two or more programs may be run in parallel, thus enabling the use of multi-processing capabilities of the computer without the need for parallel coding.

The CLI can be used to run tools on the command line. However, the real power of the CLI is that it can be used to create scripts that run a sequence of tools in order to run a complete processing chain. Furthermore, scripting can also be used for time-consuming tasks that run for a long time or automate the processing of large sets of images.

Most of the functionality of HypPy can also be imported into other Python programs using the API.

The GUI of a HypPy tool consists of a separate module, which imports the HypPy function, creates the GUI, and passes the parameters to the HypPy function. One module contains the GUI, while another module contains the API and the CLI. For instance, the `edgy.py` module contains the functions for hyperspectral edge filtering plus the command-

line interface. The GUI is invoked using the tkEdgy.py module. In this case, the three levels are as follows:

- GUI program: tkEdgy.py;
- CLI program: edgy.pyl
- API function: edgy().

The HypPy top-level menu can be used to start the GUI program tkEdgy.py. The Edgy filter can be found as a ‘Spatial–Spectral Edge Filter’ in the ‘Filter’ menu.

2.1.1. GUI

Most HypPy tools are split into two programs, one for the GUI and one for the API and the CLI. Most GUIs are simple interfaces for setting input and output parameters, which are passed to the API. An example of a GUI can be found in Section 2.4.1, where the Band Math Tool is discussed.

The user interfaces for the GUI processing tools are created using Tkinter, the GUI (graphical user interface) package that is part of the official Python distribution. Tkinter is a thin object-oriented layer on top of Tcl/Tk [10]. There are many packages that can be used to develop GUIs in Python. For instance, wxPython is a more extensive package (e.g., a widget set) than Tkinter. However, the Tkinter module (Tk interface) is the standard Python interface to the Tk GUI toolkit [11] and was chosen for HypPy for its simplicity and availability. All the names of the HypPy tools start with ‘tk’ to indicate that these have a GUI.

2.1.2. CLI and Scripting

The command-line interface is useful for running processes at the command prompt. However, the most important reason to have a CLI is to be able to run a processing chain from a script. Scripts are convenient for repeating a set of commands in an iterative manner. In this way, a potentially large set of images can be processed in exactly the same way. Running scripts reduces errors and offers the possibility to run commands that require a lot of processing time overnight.

HypPy adheres to the conventions used to supply command-line options and input and output on the command line. For that purpose, it uses the Python standard module called argparse, which offers a convenient way to build command-line interfaces that can be used in Unix as well as Windows shells.

2.1.3. The API

An application programming interface (API) is a software interface by which one piece of software offers functionality to other software. In Python, functionality from one program can be imported into another program by making use of the import statement. An important part of the API is the API specification, which describes the functions and data structures that are needed to make use of the available functionality.

Currently, a dedicated API manual is not present in HypPy, but essential classes like Image and Spectrum are covered in the Programmers’ Manual and the Spectral Math Manual, respectively. Additionally, many HypPy functions are described in the HypPy Scripting Manual (CLI) and HypPy User Manual (GUI), as the CLI and GUI often act as a thin shell around these functions.

Because data objects are an important part of the API, HypPy’s header and image objects are discussed below; see Section 2.2.

2.1.4. The Top-Level Menu

The top-level menu of HypPy can be configured using the menu configuration file called hypppymenu.cfg. Functions can be Python scripts (typically the HypPy GUI programs), PDF or HTML files, or complete URLs. In this way, the top menu can be used to start scripts as well as present documents such as manuals or web pages.

The tools are executed as separate processes; therefore, multiple HypPy tools can be started and executed in parallel at the same time. Figure 1 shows the top menu of HypPy.

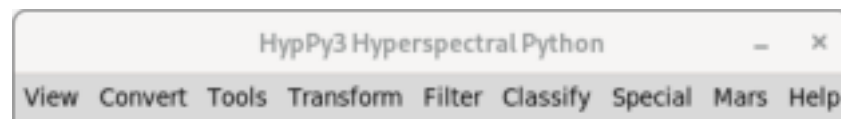


Figure 1. The top-level menu of HypPy.

Python scripts are executed by Python as subprocesses. The PDF, HTML, or URL is delegated to the default reader of the system, which will start the PDF reader or web browser on the system.

2.2. Image Formats

HypPy has adopted the ENVI formats as its native formats. The ENVI image and ENVI spectral library formats are binary files with an accompanying text header file. ENVI supports three different types of interleaving in its native format: band sequential (BSQ), band-interleaved by pixel (BIP), and band-interleaved by line (BIL).

HypPy supports all three ENVI formats: BSQ, BIP, and BIL. In addition, HypPy can map an ENVI spectral library onto an image. The spectral library is then represented as a y by x by z image, in which y is the number of spectra stored in the spectral library, x is always 1 (the spectral library is visible as an image with just one column of pixels), and z is the number of bands in the spectral library. The advantage is that spectral libraries can be viewed just like spectral images. On top of that, a processing chain meant for spectral images can be tested on a spectral library first to see what effect the processing chain will have on the various minerals present in the spectral library.

For consistency in coding, single-band images are mapped onto three dimensions as well, in which case the band dimension is always 1. There is one exception to this rule: classification images are mapped as two-dimensional one-band images. Classification images have a different use and semantics than hyperspectral data, and for this reason, classification images are treated in a different way in HypPy.

Internally, HypPy maps all three formats, BSQ, BIP, and BIL, onto an array with a BIP structure, thus relieving the programmer of having to deal with three different formats.

2.2.1. Header and Image Objects

HypPy's header module is used to read and write ENVI header files. ENVI images and spectral libraries typically consist of two files, the data and the header. The data files contain raw binary data. The header files contain human-readable text.

Inside HypPy, the ENVI header is represented as a Python object of the class Header. Objects of this class contain attributes that reflect the attributes as they are found in the ENVI header files. In the Python code, there are a few differences. First, the names of the attributes in the ENVI header can contain spaces, which is inconvenient in a programming language like Python because variable names are not allowed to contain spaces. A choice was made to replace the spaces in ENVI attribute names with an underscore. For instance, the 'data-type' attribute in the ENVI file header will be renamed to 'data_type' in Python. Second, some of the values inside the ENVI header may change their appearance in Python. For instance, IDL records, as they appear inside ENVI headers, such as values of the wavelength attribute, are changed into the standard Python list data type. The translation between IDL records and Python lists is handled by HypPy internally.

HypPy supports all IDL/ENVI raw binary data types from 8-bit unsigned integers to 64-bit signed integers and from 32-bit single floating point numbers to 64-bit double-precision complex numbers with a real-imaginary pair of double precision. Furthermore, through the use of the 'byte order' field name, HypPy supports both big-endian and little-endian byte order files.

HypPy has two factory functions in its image module, the `Open()` function for opening existing images, and the `New()` function for setting up new (empty) images.

The factory function `Open()` returns one of the image objects of class `ImageBIP`, `ImageBIL`, `ImageBSQ`, `Classification`, or `ImageSL` (a spectral library mapped as an image). Images accessed using the `Open()` function can only be read, not written. To protect existing images, these are always opened as read-only.

The factory function `New()` returns one of the image objects of class `ImageBIP`, `ImageBIL`, `ImageBSQ`, or `Classification`. In addition to these classes, the class `ImageSL` is also supported, which is an image representation of an ENVI spectral library.

2.2.2. Memory Map

HypPy uses Numpy's memory map, '`numpy.memmap`', to map binary data files onto arrays. Memory-mapped files are used to access large files on a disk without having to read the entire file into memory first. NumPy `memmap` objects are array-like objects; they can be manipulated in the same way as numpy arrays. Because, in HypPy the choice was made to have the BIP structure for the internal array, the axes of the two other formats, BSQ and BIL, must be swapped to look like BIP in the returned memory map. This is achieved by calling the transpose function of the memory map. BSQ is mapped using a `transpose(1, 2, 0)`. BIL is mapped using a `transpose(0, 2, 1)`. The BIP format does not need to be transposed.

One would think that using the transpose function would create a copy of the data with the axes of the data swapped in a different way. However, this is not the case because, for certain operations, numpy supports what is called a 'view' on the data. A view creates a reference to an existing array, with the required reorganization of the data added to the metadata. This has two advantages: first, the view does not create a copy of the data in memory, and second, when the view is opened in read-write, any modifications to the view will be reflected immediately in the original data.

Furthermore, endianness and byte order can be changed by creating a view. The three formats (BSQ, BIP, and BIL), endianness (little endian and big endian), and data type (from 8-bit unsigned integers to 64-bit floating points, etc.) can be handled by creating a view of the data on file. For the input and output image data, no copy needs to reside in the memory of the computer. In such a way, HypPy can handle large files that would normally not fit in memory.

The following functions are applied to create views:

- Creating a memory map:
`numpy.memmap(fname, mode = 'r', shape = shape, dtype = data_type);`
- Changing the byte order: method `newbyteorder()`;
- Swapping the axes from BSQ to BIP: method `transpose(1, 2, 0)`;
- Swapping the axes from BIL to BIP: method `transpose(0, 2, 1)`.

Unfortunately, changing the data type of an input file creates a copy in memory, not a view:

- Change the data type of the input: method `astype(dtype)`.

However, there are a few caveats in the use of numpy memory maps. Not all operations and functions that can be used on numpy arrays can be used on memory maps, and only a limited number of operations or functions on a memory map are capable of returning a view.

Additionally, the use of a memory map itself has disadvantages as well. Setting up a memory map uses a bit of extra memory, and all the reorganization of a view must be executed on-the-fly whenever the data on file are accessed. In this way, the memory map is less efficient than making an array copy in memory.

The advantages of a memory map are obvious, it enables the handling of large datasets that cannot possibly fit in the physical memory of the computer. Furthermore, internally, the data are mapped on the same BIP structure all the time; in a program, there is no need

to keep track of the real interleaved format of the data file. All this is handled transparently by the memory map and the views thereof.

2.2.3. Spectral Library Formats

HypPy supports two basic types of spectral libraries: ENVI spectral libraries and ASCII or text-based spectral libraries. In ENVI spectral libraries, a major drawback is that the list of wavelengths has to be the same for all the spectra in the spectral library. Therefore, the number of bands and the central wavelengths of all the spectra must be the same for all included spectra. An advantage of this format is that it is a binary format and can be addressed like an image, which offers the possibility to process spectral libraries in the same manner as spectral images. This can be used to test processing chains on spectral libraries before running them on images. A process for detecting certain minerals can be tested on the USGS spectral library [12], for instance, before applying it to an image. Problems with false positives of certain minerals can be caught in the early stages of developing a processing chain.

On the other hand, the ASCII spectral libraries offer more flexibility. Text files can be imported into other programs, and the spectral ranges of the individual spectra do not need to be all the same. A complication is that spectra must be resampled before use. HypPy resamples spectra on-the-fly, using the `interp1d()` function from the `scipy.interpolate` module to unite both sets of wavelengths. An intermediate step of resampling spectra is not needed.

2.2.4. Bad Band Lists

The ENVI image format supports the notion of what is called a bad band list (BBL), indicated in an ENVI header file by the keyword BBL. There must be as many items in the BBL as there are bands in the image. Sequentially, the bands are marked as bad or good. A zero (0) for a particular band signifies that the data in this band should not be trusted because of a sensor failure or excessive noise. A one (1) for a band signifies valid data. HypPy can take into account the bad band list, such that the bad bands do not show up in array data. In most HypPy tools, this can be indicated by setting the `use_bbl` option to true. If this option is set to false, then all of the data are visible in the array, including the bad bands. Such a virtual image, on which the `use_bbl` is set to true, may have fewer bands than the original data file.

Internally, this is handled by an index-to-index array, which is used as an indexing array to access multiple elements at once. The disadvantage is that handling such virtual images is slower than using the full image. Another disadvantage is that the band numbers in the virtual image are not the same as the band numbers in the original image. However, one of the design ideas of HypPy is to use the wavelength rather than the band number to address a spectral band of an image (see next section).

2.2.5. Sorting Wavelengths

Often, sensors consist of a number of detector banks, each covering a specific wavelength range. The data from these detectors may have overlapping wavelength ranges or may even appear in an unexpected order. For instance, the raw data from the OMEGA instrument of the European Mars Express have the data from the SWIR1 (0.93 μm to 2.69 μm) and SWIR2 (2.53 μm to 5.09 μm) detector banks before the data from the VNIR detector (0.36 μm to 1.07 μm) [13]. Another well-known hyperspectral instrument, AVIRIS, consists of four spectrometers with overlapping ranges [14]. Note that redundant bands in the spectral overlap of the AVIRIS spectrometers can also be removed or put in the bad band list; see, for instance, [15].

Obviously, the switched order and overlapping ranges may lead to problems when plotting or processing such data. Therefore, one of the design criteria of HypPy was to be able to sort the wavelengths of a hyperspectral data cube on-the-fly. In a trade-off between memory use and CPU use, HypPy does not convert the input image into memory; therefore,

it must convert the image on-the-fly. Many HypPy tools offer the ‘sort_wavelengths’ option when opening an image.

HypPy sorts the wavelengths by keeping an index-to-index table as an attribute of the images objects. This table translates the band numbers of an input image into bands that are sorted by wavelength. The advantage is that, internally, the data look like they have their bands sorted by wavelength. The obvious disadvantage is that the band numbers of the input image may not correspond to the band numbers of the original image object.

However, if bands are addressed by wavelengths rather than by band numbers, then this last point is irrelevant. Wavelengths have physical meaning and should be the preferred way of addressing image bands. Band numbers may change and even have a different meaning from one remote sensor to another. Unfortunately, not all images have wavelengths, either because the wavelengths are missing or the images do not contain spectral data, which is why band numbers can still be used in HypPy.

2.3. Viewers

2.3.1. Image Viewer

The image viewer of HypPy offers a no-frills display for viewing hyperspectral images and retrieving spectra. The image viewer was designed for a quick interactive assessment of image quality and the retrieval of endmember spectra. Its simplicity makes it easy to use for novice users of hyperspectral data. Sliders can be used to change the display from one band to another. In color mode, three sliders are available, one for each of the RGB channels of the display. A status bar at the bottom of the window presents information on wavelengths of bands, cursor position, and more. Using the sliders and the left mouse button, the bands and the spectra of the input image can be assessed on quality and information content. Figure 2 shows the hyperspectral image of a rock sample. This particular rock sample 15-91-FE is part of the Fen Complex in Norway, which comprises a carbonatite intrusion [16].

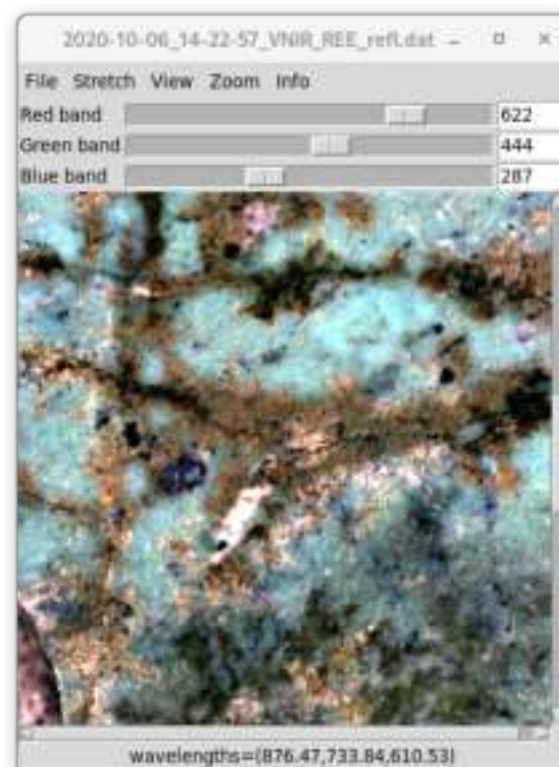


Figure 2. Image viewer showing a false color composite of sample 15-91-FE. Red channel 876 nm, green 734 nm, and blue 611 nm. The sliders for the red, green, and blue can be used to change the bands for the RGB color channels.

Under the file menu, two options can be set to control whether the bands must be sorted prior to reading or to use the BBL of an image.

The display program can open two images at the same time. The first image is always visible in the viewer. The second image, called the ‘shadow image’, is never visible. However, when the shadow image is present, spectra are obtained from the shadow image rather than the visible image. This is like a GIS in which the first layer is on top, and the second layer is hidden below the first image. This option is useful for loading a classification image or mask image in the first layer while spectra are taken from the shadow image, making it possible to assess whether a calculated classification or mask image makes sense. Furthermore, this option is useful for exploring a wavelength map together with a hyperspectral image from which the wavelength map was derived.

The stretch menu offers a number of standard methods for stretching the image [17]. Methods like 1–99% and histogram equalization are implemented. On top of that, the image can be inverted, saturation-enhanced on-the-fly, or pseudo-colored with a number of available color schemes.

A left-click anywhere on the image will bring up a plot window with a graph representing the spectrum of the pixel at this location, see Figure 3. Further clicking on the image will bring up more graphs in the plot. By removing the plot window, the process starts all over with a clear plot window. In this way, the spectra of different points in the image can be visualized together.

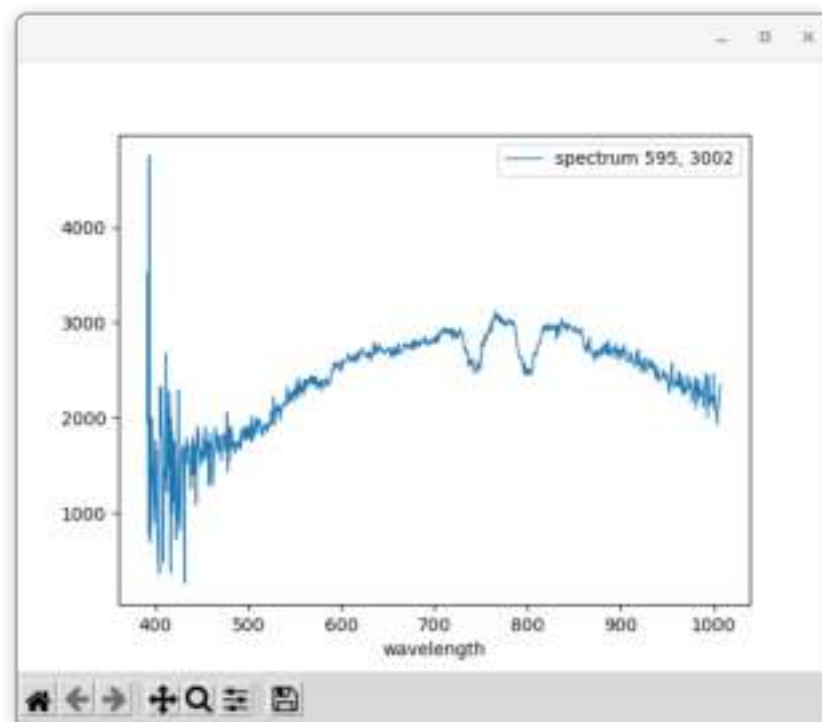


Figure 3. Plot window showing the spectrum of pixel location 595, 3002. The spectrum shows two absorption features at 740 and 800 nm. These features are due to rare earth elements (in particular neodymium) contained by monazite crystals. The spectrum below 450 nm is noisy due to the low sensitivity in the blue of the hyperspectral VNIR camera. The toolbar at the bottom of the window is a standard matplotlib figure toolbar and has buttons for navigating the figure: home, back, forward, pan/zoom, zoom-to-rectangle, and subplot configuration. The save button on the right can be used to save the figure to various formats such as png or pdf.

Using the ‘Values Window’, these spectra can also be saved as a text file, endmembers, or spectral libraries, or imported into other software for visualization and further processing. A directory of endmember spectra can be used as a spectral library in HypPy’s classifiers, spectral angle mapper, and linear unmixing.

The HypPy image display program uses the Image and ImageTk modules from the Python Imaging Library (PIL, now called Pillow [18]) for displaying images and matplotlib for making 2D graphs of spectra.

2.3.2. Spectral Library Viewer

The spectral library viewer is a viewer for ENVI-type and ASCII-type spectral libraries. After selecting the input spectral library, the names of the spectra contained by the spectral library are shown in a list. By double-clicking on a name, a plot window will pop up showing the spectrum of the selected mineral or material. Selecting multiple spectra will plot these spectra together in the plot window, enabling a comparison between spectra.

The spectral library viewer supports various ASCII formats, the CRISM spectral library format (PDS), USGS SpecLib version 6 and 7 [12], the ASD format [19], and plain-text two-column data. The spectral library viewer can recursively read an entire directory tree. See Figures 4 and 5, which show the interface of the spectral library viewer and a plot window showing an example spectrum of monazite.

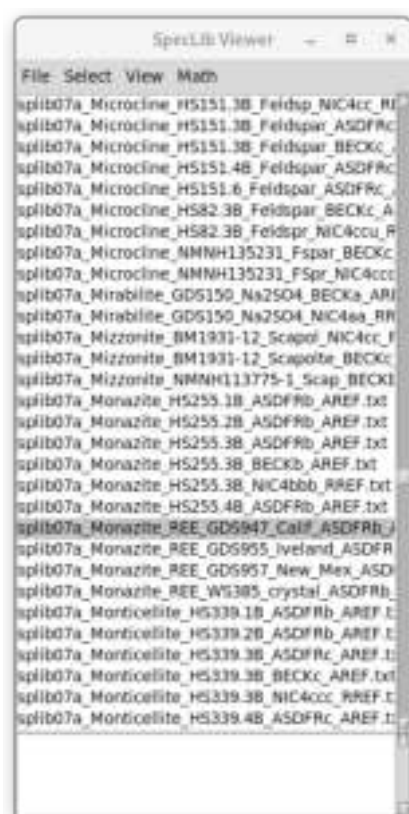


Figure 4. Interface of the spectral library viewer, in this case, loaded with the spectral library version 7 from the USGS [12].

Furthermore, the spectral library viewer supports the manipulation of spectra using a ‘spectral math’ expression. A subset of numpy functions can be applied to spectra. Indices and slices can be used on spectra, and a host of mathematical operators and functions are available for building processing operations on spectra and testing these on (a selection of) a spectral library.

Spectra with different wavelengths and wavelength ranges will automatically be resampled on-the-fly. All these functions are available in the API and are implemented in the spectrum class of HypPy (see also Section 2.4.2 below). Results of the spectral math operation in the spectral library viewer can be saved by setting an output directory first.

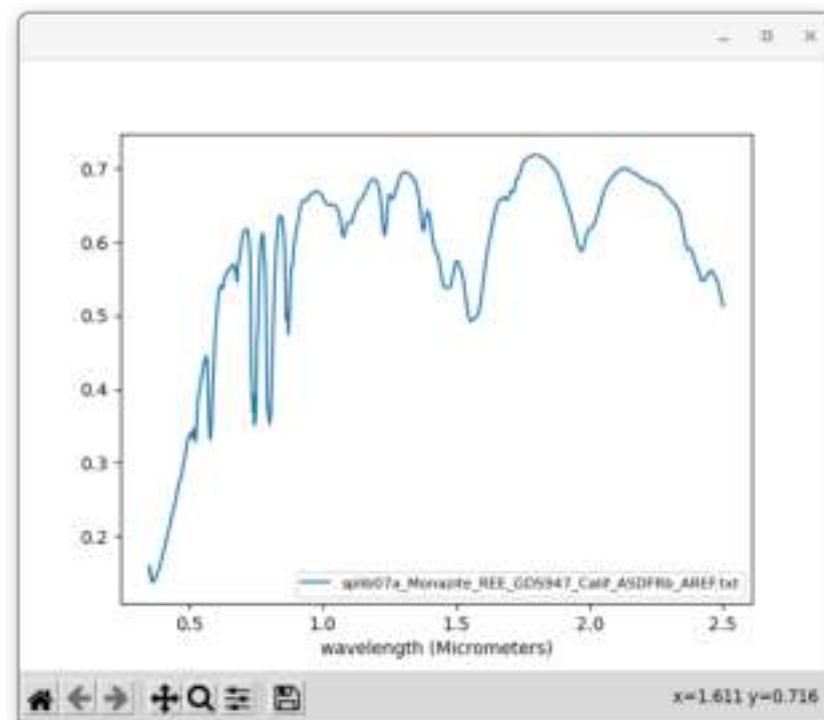


Figure 5. Plot window showing the spectrum Monazite_REE_GDS947_Calif_ASDFRb_AREF. This sample is known to contain Ce, La, Nd, and Th. Notice the Nd absorption features at 740 nm and 800 nm.

2.4. Tools

In the following, a subset of HypPy's tools will be discussed in more detail.

2.4.1. Band Math Tool

Multispectral and hyperspectral images can be regarded as a set of spectral bands, and functions are often created to process such data in a band-by-band fashion. The band math tool implements this.

An example of such a band math operation is the calculation of the NDVI, in which the difference between a near-infrared band and a red band is normalized by dividing by the sum of the near-infrared band and the red band. For the NDVI, usually for the red band, a range of 400 nm to 700 nm is used, and for infrared, the range is 700 nm to 1100 nm. In the following example, for red, we pick a band around 675 nm, and for infrared, a band around 1000 nm is chosen. In HypPy, the NDVI of an AVIRISNG image can be calculated using one of the following expressions: (1) or (2).

$$(i1[59] - i1[124]) / (i1[59] + i1[124]), \quad (1)$$

$$(i1(675.0) - i1(1000.0)) / (i1(675.0) + i1(1000.0)). \quad (2)$$

In band math, the sequence of image file names is linked to variable $i1$, $i2$ and so on. In the example of the NDVI, only one input image is needed, which is linked to the variable $i1$. Images are numbered starting at 1, and band numbers are numbered starting at 0. In the example, band 59 is the band with a center wavelength of 676.9 nm and band 124 has a center wavelength of 1002.4 nm. Please note that, in this example, the BBL is used, and the first seven bands, which are marked as bad, are not present.

In band math, it is also possible, and even preferred, to use wavelengths instead of band numbers. The band with the nearest central wavelength will be automatically selected. Note that, in band math, when wavelengths are used rather than band numbers, these wavelengths must be enclosed by round brackets, not square brackets. In Python,

this is implemented using a ‘call’ on the image class. Python objects can be designed to behave like lists or iterators, having an index between square brackets or a function with a parameter list between round brackets. Figure 6 shows an example of a band ratio.

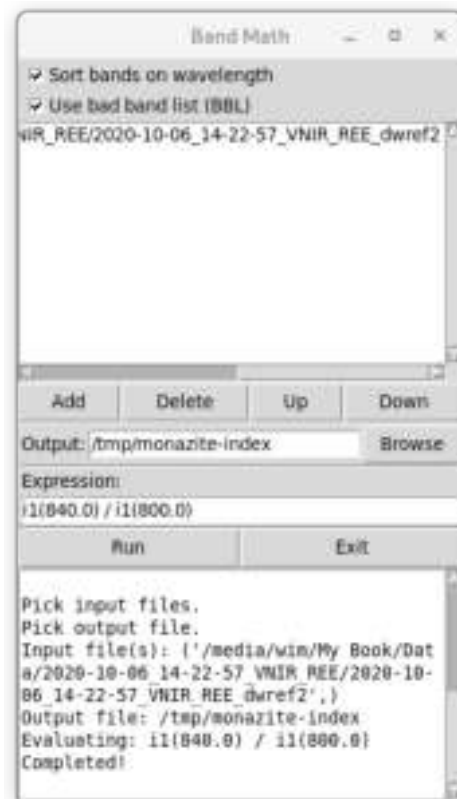


Figure 6. Interface of the band math tool. The formula shows an example of a band ratio for a simple monazite index based on the shoulder and the absorption feature of Nd at 840 nm and 800 nm, respectively. In this case, there is only one input file *i1*.

HypPy uses the Python `eval()` function to execute expressions like the NDVI example above. Band math expressions are basically Python expressions. HypPy sets up the variables *i1*, *i2*, and so on. These are HypPy image objects, which can be called to return bands of a certain wavelength. Linking the image variables to image files is an essential step in the use of band math expressions. Note that the syntax used in band math differs from the one used in spectral math. Band math uses image objects, while spectral math uses spectrum objects; see below.

2.4.2. Spectrum Class and Spectral Math

Spectral math is used to apply mathematical expressions to spectra of images or spectral libraries. Band math uses the band paradigm; spectral math uses the spectrum paradigm. In spectral math, the image is processed pixel by pixel. In band math, the image is processed band by band. Spectral math relies on the functionality as it is implemented in the spectrum class. The interface of the spectral math tool is similar to the band math tool (see Figure 6), but the expression must follow the syntax of the spectrum class.

The spectrum class was created for the manipulation of spectra. The idea is that a spectrum object contains information about wavelengths as well as values, be it reflectance, radiance, or other data. The spectrum class offers access to the data, as well as functions to operate on the spectral data.

The list below summarizes some of the properties of the spectrum container object:

- The spectrum object contains wavelengths and values.
- The spectrum object may contain a name and a description.

- An index can be used to select values or make subsets of the spectrum.
- An integer as index denotes a band number; a float denotes a wavelength.
- Mathematical operators like $+$, $-$, $*$ and $/$ can be used on spectra.
- The spectrum object contains a wealth of operators and methods. Currently, some 150 functions and 20 operators are implemented (see HypPy's spectral math manual).
- When working with spectra with different wavelengths, the spectra are automatically resampled on-the-fly.

As shown previously, spectral indices can be implemented using band math. A spectral index is supposed to have a high value if a certain mineral or material is present and a low value if it is absent. An example of such an index was given in the previous section in which we presented the example of the NDVI.

However, if the image contains noise, then the resulting NDVI may also contain noise. In such a case, it may be beneficial to work with average values in band ranges rather than single bands. In band math, it is not possible to use band ranges because bands of a range must be addressed one by one. However, in spectral math, a spectral range can be obtained using a slice on the spectrum.

The spectral math expression for NDVI may now look like the following expression (3):

$$\frac{(S1[700.0 : 1100.0].mean() - S1[400.0 : 700.0].mean())}{(S1[700.0 : 1100.0].mean() + S1[400.0 : 700.0].mean())}, \quad (3)$$

in which $S1$ denotes the spectra from the first image. Spectra of subsequent input images are numbered $S2$, $S3$, and so on. However, in this example, only one image is used. The square brackets contain what in Python are called slices, denoted by the use of a colon $:$. Notice that floats are used as indices, which denote wavelengths. Integers as indices are interpreted as band numbers. The result of a slice on a spectrum is a new spectrum object that contains a subset of the original spectrum object. In this case, the subset will contain the spectral values roughly between 400 nm and 700 nm and between 700 nm and 1100 nm.

The wavelengths are approximate because these wavelengths are matched to the closest band numbers, which may not have the exact same wavelengths as central wavelengths. HypPy does not generate errors or warnings for the conversion of wavelengths to bands to prioritize speed over quality checks, as the checks would consume significant CPU time. However, users and programmers can verify the requested and obtained wavelengths using the functions `wavelength2index()` and `index2wavelength()`.

Subsequently, the averaging function `mean()` is used on these subsets. In other words, the result will be the NDVI calculated from the average spectrum between 400 nm and 700 nm and the spectrum between 700 nm and 1100 nm. In this way, spectral indices that operate on ranges of wavelengths rather than single bands can be built. The obvious advantage is that this reduces noise in the output. The downside is that such expressions are slower than band-by-band operations. Furthermore, in the example above, the slices and means are calculated twice because the spectral math tool only supports one single expression. However, by making clever use of the walrus operator $:=$, which was introduced in Python version 3.8 [20], double calculation can be prevented. The expression for the NDVI would then look like expression (4):

$$\frac{((ir := S1[700.0 : 1100.0].mean()) - (red := S1[400.0 : 700.0].mean()))}{(ir + red)}, \quad (4)$$

in which the mean of the infrared range is stored in the variable *ir* and the mean of the red range is stored in the variable *red*, respectively, and these values are retrieved later in the expression for calculating the denominator.

Please note that we use a broad range for red and infrared in Equations (3) and (4), using the chlorophyll absorption in the visible part of the spectrum (400 nm to 700 nm) as *red* and the reflection by the leaf cell structure in the near-infrared part of the spectrum

(700 nm to 1100 nm) as *ir*. The specific wavelength ranges may vary depending on the remote sensing sensor or technology in use. For instance, Tucker [21] used the range 630 nm to 690 nm for *red* and 750 nm to 800 nm for *ir*, which are the ranges of the Landsat MSS spectral bands 5 and 7, respectively.

HypPy uses the Python `eval()` function to execute expressions like the averaging NDVI example above. The spectral math expressions are Python expressions. However, HypPy sets up the variables *S1*, *S2*, and so forth, which are HypPy spectrum objects which can be used as a Python container class. In that sense, the behavior of the spectrum object is similar to a Python list object, which can be indexed, sliced, looped over, and so on.

In HypPy, spectral math can be used in the spectral math tool as well as in the spectral library viewer.

2.4.3. Spatio-Spectral Filters

Spatio-spectral filters take into account both spectral information and spatial information. HypPy has three types of spatio-spectral filters: gradient filters, mean filters, and binning filters. Gradient filters are implemented by regarding a local spatial kernel of spectra as vectors and to calculate the weighted difference between these vectors. The mean filters calculate the weighted sum of these vectors. Binning filters reduce the number of samples by aggregating values.

HypPy has a tool for hyperspectral edge filtering. Such filters can be used to detect spectral variability in a scene. Areas with a high value are spectrally inhomogeneous, and areas with low values are spectrally homogeneous. The results of such filters can serve as a first step in image quality assessment, classification, or segmentation. The result of the gradient filters is always a single-band grey-level image. The HypPy tool called hyperspectral gradient implements a number of directional (up, down, and two diagonals) and non-directional filters (edgy and Sobel). A number of distance measures have been implemented that can be used in conjunction with these filters.

The spectral angle (SA) distance measure only takes into account the spectral information of the input image because the spectral angle is insensitive to the intensity of the input spectrum. The intensity difference (ID) distance measure, on the other hand, only takes into account the intensity information of the spectrum, not the spectral information. The Euclidean distance (ED) measure is sensitive to both the spectral and intensity information of the spectra, although most of the time, the ED seems to be dominated by the intensity of the input spectra. In addition to the standard spectral difference measures discussed above, two other distance measures, which are widely used in hyperspectral analysis, are available: the Bray–Curtis (BC) distance and the spectral information divergence (SID) [22]. All these distance measures have their own properties regarding spectral similarity.

Where the gradient filters take into account a 3×3 two-dimensional set of full spectra, the mean filters take into account a smaller three-dimensional set of spectral values. The mean filters apply a $3 \times 3 \times 3$ kernel on the hyperspectral data cube. The filter can take all the values of the kernel into account or a limited set of the kernels. Current kernels in HypPy calculate the weighted average of 27, 19, or 7 of the values of the kernel. Two averaging functions are implemented: the mean and median. A mean filter is a linear filter and, therefore, predictable in what the effect on the image will be. A median filter reduces the effect of outliers. In that respect, a median filter may be useful for filtering out pepper-and-salt noise. However, the effect of a median filter is not as easy to predict as a mean filter, which may lead to unexpected effects in the output image. In certain cases, median filters may have an effect comparable to morphological filters in the sense that areas seem to shrink or grow, which may be confusing and thus complicate the interpretation of the resulting image.

The result of a spatio-spectral mean filter is always a hyperspectral cube with the same dimensions as the input. However, like with regular two-dimensional spatial filters, edge effects should be taken into account. The first and last values in any of the three dimensions (*x*, *y*, and *band*) are treated differently than the rest of the hyperspectral cube.

In addition to these filters, HypPy has a number of binning tools, spatial, spectral, and spatial–spectral binning, to reduce the number of samples in the image cube. These tools can be used to decrease the data volume, reduce the noise by averaging, and speed up subsequent processing steps. For instance, spatial–spectral binning of $xy = 3 \times 3$ and $z = 11$ will reduce the data volume by almost a factor of 100 ($3 \times 3 \times 11 = 99$) and reduce the standard deviation of the noise by almost a factor of 10 ($\sqrt{99}$).

2.4.4. Wavelength Mapper

Mapping physical properties from hyperspectral images is complicated and often relies on the availability of spectral libraries of target materials. Many existing methods depend on the subjective decisions of the interpreter for the selection of endmembers and the post-processing of rule images. If no prior information is available, information should be extracted by characterization of spectral absorption features. On multi-spectral images, such features can be found using indices based on band ratios and band depths; see, for instance, [23]. However, hyperspectral images show more spectral detail of the features, which opens up the possibility of using more sophisticated methods.

Absorption features of spectral curves are important for analysis and interpretation. Such features may be parametrized as center wavelength position, depth, area, and asymmetry [24]. Usually, the most important features are the center wavelength position and the depth of the main absorption feature. The wavelength position of an absorption feature can be used for the identification of minerals because many minerals have their own unique absorption wavelengths. The depth of such a feature is an indication of the abundance of a mineral.

The center wavelength and depth of the deepest feature and consecutive deepest features can automatically be determined. These parameters can be calculated for all the pixels (spectra) of a hyperspectral image and presented as grey-level images representing the wavelength and depth of the feature.

Subsequently, the resulting wavelength and depth images can be fused using an HSI (hue–saturation–intensity) transform [17] in such a way that the resulting image shows the wavelength in color (the hue). The intensity represents the depth of the feature. The saturation is set to 1. The wavelength mapper tool of HypPy automatically determines the wavelength and depth of the deepest absorption feature of all the pixels of a hyperspectral cube and fuses the two into one color image.

Figure 7a shows a rock sample in a natural color composite. Figure 7b shows the result of the wavelength mapper; Figure 7c shows the legend to the wavelength map. The REE-containing monazite is visible as orange grains. The greens and the blues are carbonates. For a further discussion of this rock sample, see [16].

The wavelength mapper is unsupervised, straightforward, and repeatable; large areas can be mapped at once, and details are preserved within the focus of the chosen spectral range. The result, called the wavelength map, can be understood by users with a limited image processing background. However, although the colors may suggest the presence of certain minerals, the resulting wavelength map is not a mineral map. The wavelength map can be used as a pre-classification step in order to understand the spectral variability in the scene. The knowledge obtained from the wavelength map is useful for the additional steps of producing a mineral map.

The wavelength mapper relies on only two input parameters: the wavelength range to focus on, and the stretch to use for the depth of the feature. The wavelength mapper was first published in van Ruitenbeek et al. [25], and a tutorial on the method was published in [24].

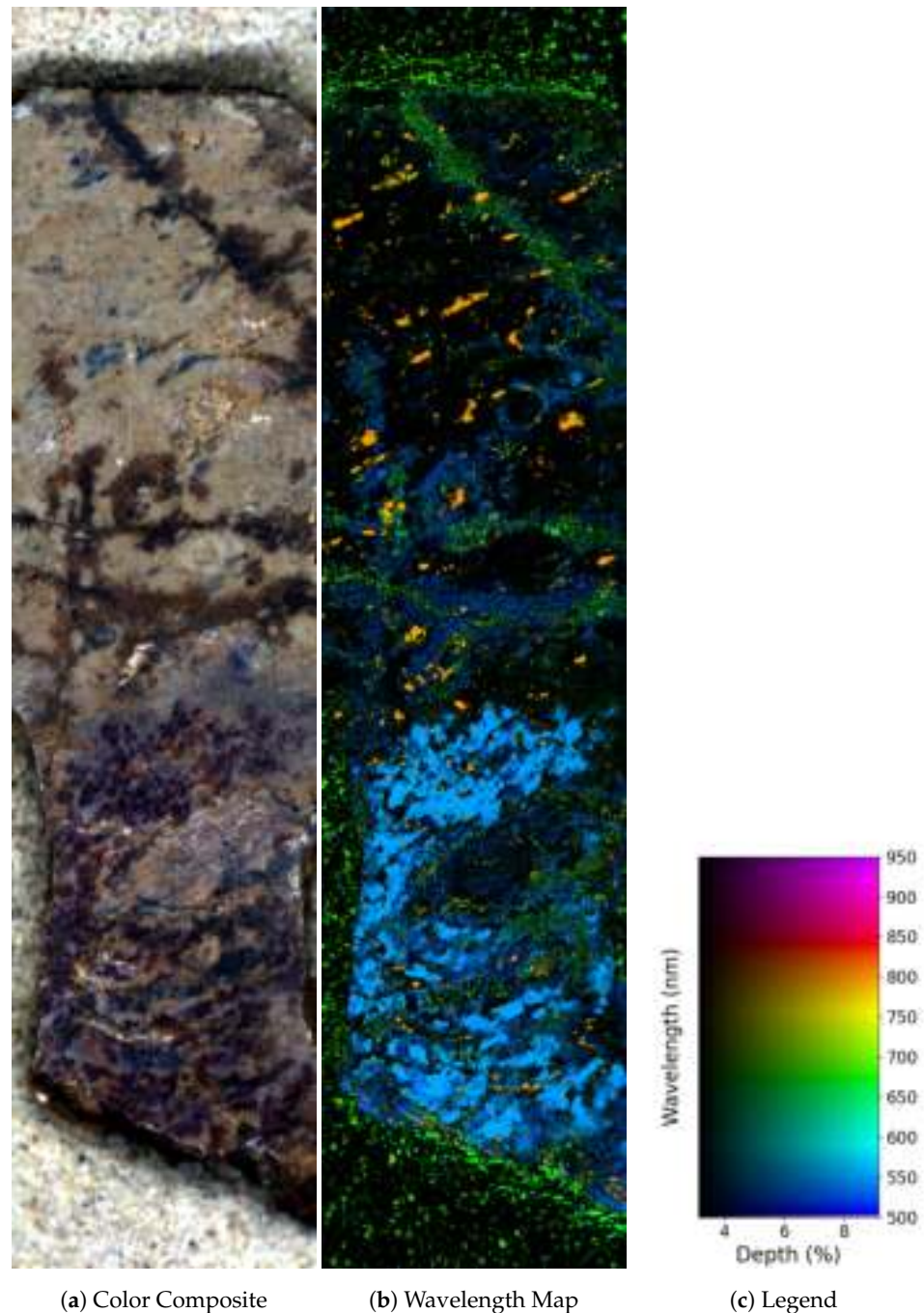


Figure 7. (a) Natural color composite of rock sample 15-91-FE (red 650 nm, green 550 nm, and blue 450 nm). (b) The wavelength map of the range of 500 nm to 950 nm shows the wavelength and depth of the deepest feature. Monazite is visible in orange (showing the 800 nm feature of Nd) or yellow–green when the 740 nm feature happens to be deeper than the 800 nm feature. (c) Legend to the wavelength map.

2.4.5. Decision Trees

A decision tree is a type of classifier that consists of a series of binary decisions used to determine the class for each input pixel. The decisions are based on the characteristics of the datasets. Every decision partitions data into either one of two potential classes or groups of classes. Decisions are expressions applied to a single image or a set of images. Each of these images represents a property of the input pixels.

One such property, for instance, could be the wavelength position of the deepest absorption feature of the spectrum of the input pixel. An expression could be built to check

whether the input pixel might contain a clay mineral with a deepest absorption feature of around 2.2 μm . Subsequent decisions could check the water features of this pixel to see whether the material has a high crystallinity index. After a number of decisions, a class will be assigned to the pixel.

Decision trees are useful for building classifiers that are based on physical properties rather than statistical inference. Of course, this can only be carried out if all the properties needed for the decision trees can be established with enough confidence.

A package like ENVI supports the construction and visualization of new decision trees. However, ENVI has a number of disadvantages. The way decision trees are implemented in ENVI is rather rigid because once a node in the tree is constructed, its place in the decision tree is fixed. In a finalized ENVI decision tree, it is impossible to move elements around. In ENVI, if you want to change the decision tree, you have to rebuild it from scratch.

HypPy can be used to execute ENVI decision trees. The only condition is that expressions used for decisions can be translated into expressions in Python.

In addition to the ENVI decision tree format, HypPy has its own decision tree format. This format is a text-based format that can be edited using a text editor.

For the visualization of decision trees, HypPy uses the Graphviz package [26]. To this end, the ENVI or HypPy decision tree must be converted to the 'dot' format first. The dot language is an extensive language for describing graphs. The binary decision trees used by ENVI and HypPy present a tiny subset of all possible graphs and types of graphs supported by the Graphviz package.

2.4.6. Zonal Statistics

The zonal statistics tool calculates statistics on pixels of an input image within the zones defined by the zone image. The input image is a regular hyperspectral or multispectral image. The zone image can be the result of a classification image or a mask image. For each distinct value in the zone image, an aggregation function is run on the input image for all the locations having this distinct value in the zone image. The currently implemented aggregation functions are as follows: mean, median, minimum, maximum, standard deviation, and mean plus or minus two times the standard deviation.

The results of the zonal statistics are useful for determining the properties of areas in the input image. For instance, in the case of noise in the input image, the mean spectrum of an area may reveal more spectral detail than the spectrum of a single pixel, in which spectral details of features may be drowned out by noise.

3. Application Areas

Below, the application of HypPy in several projects in education, consulting, and research will be discussed in more detail.

3.1. Education

HypPy is one of the software packages used in our spectral geology courses at the University of Twente [27]. The clearness of the HypPy interface makes it an excellent tool for students starting out in spectral geology. One example of published student research using HypPy is [28]. For a tutorial on analyzing absorption features for finding ore using geological remote sensing, see [24].

HypPy can be used in the short courses to learn the basic concepts of hyperspectral imaging and carry out specialized operations on hyperspectral data processing and interpretation. For example, HypPy was used in the course 'Remote Sensing and Mineral Spectroscopy for the Exploration and Mining Geologist' offered in Istanbul [29].

In December 2022, HypPy was used in a spectral workshop during the Annual GRSG Conference [30]. The workshop comprised an overview of the theory and practical implementation of spectral methods for mineral identification, accompanied by a hands-on laboratory session involving the application of spectral processing analytics to a recently obtained high-resolution airborne hyperspectral dataset that spans the visible and near-

infrared spectra (VNIR 400 nm to 1 μm), short-wave infrared (SWIR 1 μm to 2.5 μm), and long-wave infrared (LWIR 8 μm to 15 μm) spectral ranges. The laboratory session employed the HypPy hyperspectral processing software and provided unhindered access to sample data.

3.2. Consulting

We explored the viability of employing laboratory-based short-wave infrared (SWIR) hyperspectral imagery for a commercial partner to identify and map the mineralogical composition of specific sedimentary drill core samples from the subsurface of the Netherlands. The scripting capabilities of HypPy proved essential in this project.

Our findings indicate that utilizing laboratory-based short-wave infrared (SWIR) hyperspectral images with a spatial sampling of 200 μm offers a valuable non-destructive approach for identifying and mapping white mica, clay, and carbonate minerals in sedimentary drill cores. Together with partners at the University of Groningen, using HypPy we succeeded in building a prototype processing chain for the quantification of the mineral content and deriving a number of geophysical properties of the drill cores.

3.3. Research

The first work on HypPy started with the Mars Planetary Mapping Pilot Project for the European Space Agency (ESA). The pilot study was designed to transfer the modern, digital methods and workflows used in terrestrial geological mapping to planetary mapping. The focus of this study was placed on the ESA datasets from the High Resolution Stereo Camera (HRSC) and the Visible and Infrared Mineralogical Mapping Spectrometer (OMEGA) hyperspectral instrument onboard the Mars Express [31]. The first step of the workflow was to process the raw data to highlight features of interest in derived information products. These included digital elevation models extracted from the stereo data and mineral maps made using the hyperspectral data. These tools were used to interpret the data and produce a geological map for the Nili Fossae area. This study paved the way for the wavelength mapping technique to be adopted for mapping the planets Mars and Earth on a routine basis [7].

HypPy introduces a processing methodology designed to transform OMEGA data from radiance-at-sensor to surface reflectance. To address particular challenges associated with OMEGA data, we devised innovative approaches for determining a per-scene transmittance spectrum, conducting atmospheric correction, and filtering systematic and random noise. See, for instance, van Ruitenbeek et al. [25]. These filters are now part of HypPy. The processing methodology generates surface reflectance images, allowing the extraction of single-pixel spectra for surface feature identification. In alignment with various prior studies, our findings suggest, among other observations, the existence of iron-rich clays on the Martian surface.

Mapping physical properties from hyperspectral images is complicated and frequently depends on the accessibility of spectral libraries of the target materials. Many existing methods depend on the subjective decisions of the interpreter for the selection of endmembers and the postprocessing of rule images. If no prior information is available, information should be extracted by the characterization of spectral absorption features. In van Ruitenbeek et al. [25], they describe a method called the wavelength mapper (see Section 2.4.4) to automatically determine the minimum reflectance within a wavelength range. We conclude that the wavelength mapper proves to be a useful tool for unbiased endmember selection, mineral mapping, and the interpretation of areas demonstrating zonation of mineral alterations using hyperspectral imagery. The usefulness of the wavelength mapper has already been recognized in other research. For instance, it was used to map mineral outcrops of lithium-bearing pegmatites in a pilot quarry located near Uis, Namibia [32].

In another study, the entropy function was added to the spectrum class to enable the calculation of an index for sorting processes in hydrothermal systems. Hydrothermal processes affect the composition of rocks by chemical reactions and fluid flow, and this

paper showed that the remaining patterns of such processes can be visualized using the Shannon entropy. The application of entropy is useful for studying hydrothermal mineral deposits and may be valuable in studies of early life environments as well.

Functions that have already been implemented in the spectrum class of HypPy could be used to add noise to data. This was used for the simulation of airborne SWIR hyperspectral sensors for the mapping of methane (CH₄) gas [33]. The study's findings indicated that three presently active airborne imaging spectrometers were unsuccessful in detecting the CH₄ plume. The paper emphasized the importance of both spectral sampling and radiometric calibration for the successful mapping of methane.

Contemporary imaging spectrometers are compact, lightweight, cost-effective, and applicable in laboratory settings. Precise hyperspectral images necessitate minimal noise, precise calibration, and consistent response uniformity. Periodic checks are advisable due to the aging of optical and electronic components. In another study we introduce a technique for quantifying smile and keystone geometric distortions from test images [34]. The measurements are acquired through the utilization of straightforward models for smile and keystone, along with pre-existing tools in HypPy. Additionally, functions for quadratic spline interpolation, normalized cross-correlation, and optimization are applied to obtain the smile and keystone values.

Findings indicate that the smile and keystone of a laboratory imaging spectrometer can be determined with sub-pixel precision.

4. Discussion

In addition to HypPy, several other packages offer functionality for processing hyperspectral images.

To name a few, QUANTools was implemented using the ENVI/IDL programming language for the automatic detection of multiple absorption feature parameters [35]. However, to run QUANTools, a runtime version of ENVI/IDL is needed, and the QUANTools package offers limited functionality, similar to one specific tool in HypPy, which is called the wavelength mapper.

PySptools is a Python module that implements spectral and hyperspectral algorithms [36]. However, this is a package or library, and you have to do coding before you can use it. In fact, HypPy uses pysptools for its linear unmixing classifier. Unlike pysptools, HypPy offers a user interface to the functionality.

An additional noteworthy toolkit is known as spectral Python (SPy) [37]. SPy is a Python module designed for handling hyperspectral image data, featuring functions for tasks such as reading, displaying, manipulating, and classifying hyperspectral imagery. Notably, SPy is limited to interactive usage from the Python command prompt or through Python scripts, unlike HypPy.

The Orfeo and EnMAPBox software packages are both powerful tools used in remote sensing and image processing applications [38,39]. They are widely used in the remote sensing community and provide extensive capabilities for processing and analyzing satellite imagery. While these packages focus on the processing of satellite images, HypPy is also targeted at processing general hyperspectral images, such as proximal images recorded with hyperspectral cameras in the laboratory.

Just like the Python language itself [6], HypPy was designed with simplicity rather than performance in mind [40]. Some operations in HypPy may be relatively slow. However, the modularity of the functions and tools can be used to build experimental programs or even prototypes of processing chains that run on any hardware.

Regarding speed, future versions of Python may be faster. To name one ongoing project, the Mojo language is a superset of the Python language with the speed of the C language [41].

5. Conclusions

In this paper, we described the design, implementation, and application of the Python package called Hyperspectral Python, HypPy. The idea of HypPy is to be able to process hyperspectral images using free and open-source software. HypPy was developed using the Python language and relies heavily on the array-processing capabilities of packages like NumPy and SciPy. HypPy can be used for early data analysis and prototyping of processing chains. HypPy was designed with simplicity and practical imaging spectrometry in mind. Simultaneously, it incorporates several innovative and potent concepts.

HypPy is a package in development and for development, and has proven its merits. Its simplicity and flexible top-level menu structure makes HypPy attractive for use in education and short courses. New methods for research can quickly be programmed and tested. Examples of such research projects are the ESA Mars planetary mapping pilot project and the research project for determining the keystone and smile of hyperspectral line cameras.

In projects, it has been used to build entire processing chains from hyperspectral data to information. In short, HypPy can be used as a platform for generating and testing new ideas for the processing of hyperspectral data. The core of reading and writing hyperspectral images, the wavelength mapper, hyperspectral gradient filters, spatio-spectral filters, band math and spectral math tools with their own expression language, decision tree tools, the use of zonal statistics, and dedicated image and spectral library viewers are good examples of that.

We expect HypPy to also be of great added value to other organizations and individuals in research, education, and projects using hyperspectral data.

Author Contributions: Conceptualization, W.B. and F.v.R.; methodology, W.B.; software, W.B.; validation, W.B., F.v.R., H.v.d.W. and C.H.; resources, A.D.; writing—original draft preparation, W.B.; writing—review and editing, H.v.d.W.; visualization, W.B.; supervision, H.v.d.W. and F.v.d.M.; project administration, W.B.; funding acquisition, F.v.d.M. and F.v.R. All authors have read and agreed to the published version of the manuscript.

Funding: We acknowledge the financial support received from the European Space Agency (the Mars planetary mapping project) and Wijnand van Eindhoven of Deep Atlas. The funding provided by these organizations played a crucial role in the successful execution of our research project.

Data Availability Statement: HypPy can be downloaded from Github and Zenodo, see [42]; Name: HypPy (Hyperspectral Python); License: GPU General Public License Version 3; Contact: w.h.bakker@utwente.nl; Hardware requirements: regular windows, mac, or linux laptop or desktop; Program language: Python version 3; Minimum software requirement: numpy, scipy, pillow, matplotlib, pysptools; Program size: 12.2 MB.

Acknowledgments: We are grateful to the University of Twente for furnishing the essential resources and facilities that greatly supported the seamless advancement of our research. Furthermore, we are grateful to all students who directly or indirectly contributed to this work by using HypPy during their studies. Their collaboration and encouragement were instrumental in the shaping of HypPy.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

API	A Programming Interface
BBL	Bad Band List
BIL	Band Interleaved by Line
BSQ	Band Sequential
CLI	Command-Line Interface
GUI	Graphical User-Interface
SWIR	Shortwave Infrared
VNIR	Visible and Near-Infrared

References

1. NV5 Geospatial Solutions Inc. ENVI—Environment for Visualizing Images. 2023. Available online: https://www.nv5geospatialsoftware.com/docs/using_envi_Home.html (accessed on 24 November 2023).
2. Prokakis, E. Free and Open-Source Software: Freedom, Transparency and Efficiency in the Digitalization Era. *J. Politics Ethics New Technol. AI* **2022**, *1*, e31230. [CrossRef]
3. Harris, C.R.; Millman, K.J.; van der Walt, S.J.; Gommers, R.; Virtanen, P.; Cournapeau, D.; Wieser, E.; Taylor, J.; Berg, S.; Smith, N.J.; et al. Array programming with NumPy. *Nature* **2020**, *585*, 357–362. [CrossRef] [PubMed]
4. Virtanen, P.; Gommers, R.; Oliphant, T.E.; Haberland, M.; Reddy, T.; Cournapeau, D.; Burovski, E.; Peterson, P.; Weckesser, W.; Bright, J.; et al. SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nat. Methods* **2020**, *17*, 261–272. [CrossRef] [PubMed]
5. Hunter, J.D. Matplotlib: A 2D Graphics Environment. *Comput. Sci. Eng.* **2007**, *9*, 90–95. [CrossRef]
6. van Rossum, G.; Drake, F.L. *Python 3 Reference Manual*; CreateSpace: Scotts Valley, CA, USA, 2009. Available online: <https://dl.acm.org/doi/book/10.5555/1593511> (accessed on 31 July 2024).
7. Tragheim, D.G.; Marsh, S.H.; Pedley, R.C.; Napier, B.R.; Bateson, L.; Smith, A.G.; Marchant, A.P.; Gunnink, J.L.; Oosthoek, J.H.P.; Muller, J.P. *The Mars Planetary Mapping Pilot Project*; European Space Agency (ESA): Paris, France, 2010.
8. Konstantinides, K.; Rasure, J.R. The Khoros software development environment for image and signal processing. *IEEE Trans. Image Process.* **1994**, *3*, 243–252. [CrossRef] [PubMed]
9. Young, M.; Argiro, D.; Kubica, S. Cantata. *ACM SIGGRAPH Comput. Graph.* **1995**, *29*, 22–24. [CrossRef]
10. Ousterhout, J.K.; Jones, K.; Foster-Johnson, E.; Fellows, D.; Griffin, B.; Welton, D. *Tcl and the Tk Toolkit*, 2nd ed.; Addison-Wesley Professional Computing Series; Addison-Wesley: Upper Saddle River, NJ, USA, 2009.
11. Roseman, M. *Modern Tkinter for Busy Python Developers: Quickly Learn to Create Great Looking User Interfaces for Windows, Mac and Linux Using Python's Standard GUI Toolkit*; Late Afternoon Press: Victoria, BC, Canada, 2020.
12. Kokaly, R.F.; Clark, R.N.; Swayze, G.A.; Livo, K.E.; Hoefen, T.M.; Pearson, N.C.; Wise, R.A.; Benzel, W.M.; Lowers, H.A.; Driscoll, R.L.; et al. *USGS Spectral Library Version 7*; Technical report; USGS: Reston, VA, USA, 2017. [CrossRef]
13. Bibring, J.P.; Langevin, Y.; Altieri, F.; Arvidson, R.; Beilud, G.; Berthél, M.; Douté, S.; Drossart, P.; Encrenaz, T.; Forget, F.; et al. *OMEGA: Observatoire pour la Minéralogie, l'Eau, les Glaces et l'Activité*; European Space Agency, ESA (Special Publication): Paris, France, 2009; SP-1291; pp. 75–95. Available online: <https://sci.esa.int/s/WnjXOYW> (accessed on 31 July 2024).
14. Vane, G.; Green, R.O.; Chrien, T.G.; Enmark, H.T.; Hansen, E.G.; Porter, W.M. The airborne visible/infrared imaging spectrometer (AVIRIS). *Remote Sens. Environ.* **1993**, *44*, 127–143. [CrossRef]
15. Jia, G.; Hueni, A.; Tao, D.; Geng, R.; Schaepman, M.E.; Zhao, H. Spectral super-resolution reflectance retrieval from remotely sensed imaging spectrometer data. *Opt. Express* **2016**, *24*, 19905–19919. [CrossRef] [PubMed]
16. Marien, C.; Dijkstra, A.H.; Wilkins, C. The hydrothermal alteration of carbonatite in the Fen Complex, Norway: Mineralogy, geochemistry, and implications for rare-earth element resource formation. *Mineral. Mag.* **2018**, *82*, S115–S131. [CrossRef]
17. Richards, J.A.; Jia, X. *Remote Sensing Digital Image Analysis*; Springer: Berlin/Heidelberg, Germany, 1999. [CrossRef]
18. Driscoll, M. *Pillow: Image Processing with Python*; Leanpub: Victoria, BC, Canada, 2021. Available online: <https://leanpub.com/pillow> (accessed on 31 July 2024).
19. Malvern Panalytical Ltd. *ASD File Format—Version 8*. Technical Report, Malvern Panalytical. 2017. Available online: <https://www.malvernpanalytical.com/en/learn/knowledge-center/user-manuals/asd-file-format-v8> (accessed on 31 July 2024).
20. Hettinger, R. What's New In Python 3.8. 2019. Available online: <https://docs.python.org/3/whatsnew/3.8.html> (accessed on 24 November 2023).
21. Tucker, C.J. Red and photographic infrared linear combinations for monitoring vegetation. *Remote Sens. Environ.* **1979**, *8*, 127–150. [CrossRef]
22. Chein-I Chang. Spectral information divergence for hyperspectral image analysis. In Proceedings of the IEEE 1999 International Geoscience and Remote Sensing Symposium, IGARSS'99 (Cat. No.99CH36293), Hamburg, Germany, 28 June–2 July 1999; Volume 1, pp. 509–511. [CrossRef]
23. Hewson, R.D.; Cudahy, T.J.; Mizuhiko, S.; Ueda, K.; Mauger, A.J. Seamless geological map generation using ASTER in the Broken Hill-Curnamona province of Australia. *Remote Sens. Environ.* **2005**, *99*, 159–172. [CrossRef]
24. Hecker, C.; van Ruitenbeek, F.J.A.; van der Werff, H.M.A.; Bakker, W.H.; Hewson, R.D.; van der Meer, F.D. Spectral Absorption Feature Analysis for Finding Ore: A Tutorial on Using the Method in Geological Remote Sensing. *IEEE Geosci. Remote Sens. Mag.* **2019**, *7*, 51–71. [CrossRef]
25. van Ruitenbeek, F.J.A.; Bakker, W.H.; van der Werff, H.M.A.; Zegers, T.E.; Oosthoek, J.H.P.; Omer, Z.A.; Marsh, S.H.; van der Meer, F.D. Mapping the wavelength position of deepest absorption features to explore mineral diversity in hyperspectral images. *Planet. Space Sci.* **2014**, *101*, 108–117. [CrossRef]
26. Ellson, J.; Gansner, E.R.; Koutsofios, E.; North, S.C.; Woodhull, G. Graphviz and Dynagraph—Static and Dynamic Graph Drawing Tools. In *Graph Drawing Software*; Springer Link: New York, NY, USA, 2004; pp. 127–148. [CrossRef]
27. ITC. Master of Science Degree Programme in Geo-Information Science and Earth Observation—Academic Year 2023–2024. 2023. Available online: <https://studyguide.itc.nl/m-geo> (accessed on 24 November 2023).

28. Portela, B.; Sepp, M.D.; van Ruitenbeek, F.J.A.; Hecker, C.; Dilles, J.H. Using hyperspectral imagery for identification of pyrophyllite-muscovite intergrowths and alunite in the shallow epithermal environment of the Yerington porphyry copper district. *Ore Geol. Rev.* **2021**, *131*, 104012. [CrossRef]
29. Portela, B. Short Course—Remote Sensing and Mineral Spectroscopy for the Exploration & Mining Geologist November 7–11, 2022, Istanbul. 2022. Available online: https://www.linkedin.com/posts/brunobvportela_geologicalremotesensing-istanbul-turkey-activity-7001884993555693568-bkXG (accessed on 24 November 2023).
30. Geological Remote Sensing Group. GRSG Annual Conference & AGM 2022: Orbit to Outcrop. 2022. Available online: <https://www.grsg.org.uk/grsg-agm-conference-2022> (accessed on 24 November 2023).
31. Chicarro, A.; Martin, P.D.; Trautner, R. The Mars Express mission: An overview. In *Mars Express: A European Mission to the Red Planet*; ESA: Noordwijk, The Netherlands, 2004; Volume SP-1240. Available online: <https://sci.esa.int/documents/33745/35957/1567254632829-OverviewWeb.pdf> (accessed on 31 July 2024).
32. Booyesen, R.; Lorenz, S.; Thiele, S.T.; Fuchsloch, W.C.; Marais, T.; Nex, P.A.M.; Gloaguen, R. Accurate hyperspectral imaging of mineralised outcrops: An example from lithium-bearing pegmatites at Uis, Namibia. *Remote Sens. Environ.* **2022**, *269*, 112790. [CrossRef]
33. Scafutto, R.D.M.; van der Werff, H.; Bakker, W.H.; van der Meer, F.; de Souza Filho, C.R. An evaluation of airborne SWIR imaging spectrometers for CH₄ mapping: Implications of band positioning, spectral sampling and noise. *Int. J. Appl. Earth Obs. Geoinf.* **2021**, *94*, 102233. [CrossRef]
34. Bakker, W.; van der Werff, H.; van der Meer, F. Determining Smile And Keystone Of Lab Hyperspectral Line Cameras. In Proceedings of the 2019 10th Workshop on Hyperspectral Imaging and Signal Processing: Evolution in Remote Sensing (WHISPERS), Amsterdam, The Netherlands, 24–26 September 2019; pp. 1–5. [CrossRef]
35. Kopačková, V.; Koucká, L. Mineral mapping based on automatic detection of multiple absorption features. *EARSel EProc.* **2014**, *13*, 95–99. [CrossRef]
36. Therien, C. Welcome to the PySptools Documentation. 2022. Available online: <https://pysptools.sourceforge.io/> (accessed on 24 November 2023).
37. Boggs, T. Spectral Python (SPy) User Guide. 2020. Available online: https://www.spectralpython.net/user_guide.html (accessed on 24 November 2023).
38. Grizonnet, M.; Michel, J.; Poughon, V.; Inglada, J.; Savinaud, M.; Cresson, R. Orfeo ToolBox: Open source processing of remote sensing images. *Open Geospat. Data Softw. Stand.* **2017**, *2*, 15. [CrossRef]
39. van der Linden, S.; Rabe, A.; Jakimow, B.; Thiel, F.; Cooper, S.; Okujeni, A.; Hostert, P. Integrating Imaging Spectroscopy and GIS—Free and Open Source Image Analysis in QGIS with the EnMAP-Box 3. In Proceedings of the OSA Optical Sensors and Sensing Congress 2021 (AIS, FTS, HISE, SENSORS, ES), Washington, DC, USA, 19–23 July 2021; Optica Publishing Group: Washington, DC, USA, 2021; p. HF4E.2. [CrossRef]
40. van Rossum, G.; Fridman, L. Why Python 3.11 Is So Fast. 2022. Interview on YouTube. Available online: <https://youtu.be/TLhRuZ9cJWc> (accessed on 31 July 2024).
41. Modular Inc. Mojo—A New Programming Language. 2023. Available online: <https://www.modular.com/mojo> (accessed on 24 November 2023).
42. Bakker, W.H. wimbakker/hyppy: HypPy3. *Zenodo* **2024**. [CrossRef]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.

Article

Design of Multichannel Spectrum Intelligence Systems Using Approximate Discrete Fourier Transform Algorithm for Antenna Array-Based Spectrum Perception Applications

Arjuna Madanayake ¹, Keththura Lawrance ^{1,*}, Bopage Umesha Kumarasiri ¹, Sivakumar Sivasankar ¹, Thushara Gunaratne ², Chamira U. S. Edussooriya ³ and Renato J. Cintra ⁴

¹ Department of Electrical and Computer Engineering, Florida International University, Miami, FL 33174, USA; amadanay@fiu.edu (A.M.); ukuma003@fiu.edu (B.U.K.); ssiva011@fiu.edu (S.S.)

² Herzberg Astronomy and Astrophysics Research Center, National Research Council Canada, Penticton, BC V0H 1K0, Canada; thushara.gunaratne@nrc-cnrc.gc.ca

³ Department of Electronic and Telecommunication Engineering, University of Moratuwa, Moratuwa 10400, Sri Lanka; chamira@uom.lk

⁴ Department of Technology, Universidade Federal de Pernambuco, Caruaru 55015, PE, Brazil; rjdsc@de.ufpe.br

* Correspondence: klawr041@fiu.edu

Abstract: The radio spectrum is a scarce and extremely valuable resource that demands careful real-time monitoring and dynamic resource allocation. Dynamic spectrum access (DSA) is a new paradigm for managing the radio spectrum, which requires AI/ML-driven algorithms for optimum performance under rapidly changing channel conditions and possible cyber-attacks in the electromagnetic domain. Fast sensing across multiple directions using array processors, with subsequent AI/ML-based algorithms for the sensing and perception of waveforms that are measured from the environment is critical for providing decision support in DSA. As part of directional and wideband spectrum perception, the ability to finely channelize wideband inputs using efficient Fourier analysis is much needed. However, a fine-grain fast Fourier transform (FFT) across a large number of directions is computationally intensive and leads to a high chip area and power consumption. We address this issue by exploiting the recently proposed approximate discrete Fourier transform (ADFT), which has its own sparse factorization for real-time implementation at a low complexity and power consumption. The ADFT is used to create a wideband multibeam RF digital beamformer and temporal spectrum-based attention unit that monitors 32 discrete directions across 32 sub-bands in real-time using a multiplierless algorithm with low computational complexity. The output of this spectral attention unit is applied as a decision variable to an intelligent receiver that adapts its center frequency and frequency resolution via FFT channelizers that are custom-built for real-time monitoring at high resolution. This two-step process allows the fine-grain FFT to be applied only to directions and bands of interest as determined by the ADFT-based low-complexity 2D spacetime attention unit. The fine-grain FFT provides a spectral signature that can find future use cases in neural network engines for achieving modulation recognition, IoT device identification, and RFI identification. Beamforming and spectral channelization algorithms, a digital computer architecture, and early prototypes using a 32-element fully digital multichannel receiver and field programmable gate array (FPGA)-based high-speed software-defined radio (SDR) are presented.

Keywords: approximate discrete Fourier transform; wideband spectrum perception; wideband multibeam RF digital beamforming; low-complexity algorithms



Citation: Madanayake, A.; Lawrance, K.; Kumarasiri, B.U.; Sivasankar, S.; Gunaratne, T.; Edussooriya, C.U.S.; Cintra, R.J. Design of Multichannel Spectrum Intelligence Systems Using Approximate Discrete Fourier Transform Algorithm for Antenna Array-Based Spectrum Perception Applications. *Algorithms* **2024**, *17*, 338. <https://doi.org/10.3390/a17080338>

Academic Editor: Frank Werner

Received: 3 July 2024

Revised: 24 July 2024

Accepted: 26 July 2024

Published: 1 August 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

In Tolkien's classic *The Lord of the Rings*, an omnipresent "All Seeing Eye" senses and hitherto inspires inhabitants of Middle Earth to act based on their proximity to objects under the influence of Lord Sauron [1]. In our world, we will enable persistent and

omnipresent knowledge of the electromagnetic spectrum via sensing across space, time, and waveform through radio frequency (RF) perception and thus enable intelligent action for AI-based DSA. *We need an RF All-Seeing Eye for the spectrum.* The fusion of RF spectrum sensing with AI and machine learning (ML) enables the full perception of the spectrum. Omnipresence, which implies *everywhere*, and persistence, which implies *all the time* for information on what might be happening in the electromagnetic environment, will lead to unimaginably powerful capabilities for tomorrow's intelligent wireless systems. The spectrum is an immensely valuable and scarce natural resource; however, the exponential growth in wireless connectivity inevitably leads to running out of spectrum. The problem of spectrum shortage can in theory be mitigated by moving communication networks to the mm-wave band in the 30 GHz to 300 GHz range. Nevertheless, industry has recently learned a bitter lesson with the mm-wave. Indeed, mm-wave technology works in a laboratory environment; yet, a number of issues are present: (i) it is difficult to get it to work across real-world environments as the sharp RF beams necessary for maintaining connectivity are prone to blockages; (ii) beam-search is a tough problem to solve, specially when considering mobility aspects; (iii) power amplifiers (PAs) are inefficient and waste a lot of energy; and (iv) channel coherence times are very low, requiring faster updates of channel state information (CSI), leading to a high computational load.

1.1. FR3 Band: The Goldilocks Spectrum

On the one-hand, FR1/sub-6 GHz frequency bands—the legacy spectrum—is simply not available for expanded growth. Indeed, the 100 MHz sub-band at 3.5 GHz was sold at an eye-popping USD 29B, clearly showing the demand from the wireless industry. However, all the money in the world cannot provide you with resources that do not exist. Given that mm-wave is unlikely to lead to practical systems in the near future, the industry is turning to a middle ground. This spectral middle ground lies in the FR3 band (6 GHz to 24 GHz), affectionately known as the *upper-mid band* by industry players. In our recent work, we showed that the upper-mid band is not too low such that it is as overcrowded as the FR1 (legacy band) and is not too high that building practically scalable infrastructure is not currently viable. The FR3 band sits “just right” as it lies within a range that is favorable for scalable systems. Additionally, bandwidth is available in FR3 for the next-generation of wireless technology, provided that the tangled web of spectrum policy can be navigated [2].

1.2. A Shared Spectrum Economy

There is a pressing problem with using the upper-mid band for wireless connectivity. The 6 GHz to 24 GHz range has already been assigned for mission-critical applications, and its licenses are owned by the Department of Defense (DoD) and other government entities. Most of the DoD mission-critical sensors, including X-band radar, communication networks, science experimental bands (e.g., radio astronomy), and weather radar sensors also exist within the FR3 band. Primary users (PUs) will likely never give up their “precious” spectrum permanently; however, the future will likely necessitate DSA, where secondary users (SUs) such as commercial wireless networks can make use of the FR3 bands in a way that poses little or no threat to critical applications pursued by PUs. Although DSA can bring massive benefits and economic growth to society, such great power comes with great responsibility as compliance to sharing agreements must be ensured with non-repudiation and zero trust.

1.3. The Need for Real-Time AI Spectral Perception

“DoD determined that sharing is feasible if certain advanced interference mitigation features & a coordination framework to facilitate spectrum sharing are put in place” proclaims the 2023 National Spectrum Strategy from the White House [3]. The early detection of acts of war sits higher on the list of priorities than commercial wireless network quality of service. Although the chance of a cataclysmic event is small, it is not zero. The only way to ensure that SUs do not encroach on PU spectrum allocations the across space, time,

and waveform domains is through independent measurements, assuming zero trust to fully guarantee the compliance to spectrum sharing agreements. In addition to genuine mistakes and unintentional harmful RF interference, there are impending cyber-threats from bad actors hitherto unknown. The possibility of jamming critical wireless and defense infrastructure should not be taken lightly. Reliable, high-sensitivity measurements, with RF sensing everywhere all the time allows better resilience to such future threats from a spectrum warfare standpoint.

1.4. Spectrum Sensors with Omnipresent Perception

RF autonomy due to AI systems created using deep learning (DL) and reinforcement learning (RL) are expected to generate a deep knowledge—that is “perception”—about the sensed electromagnetic environment [4]. The National Spectrum Strategy (The White House, 11/2023) calls for initial studies in the 3.1 GHz to 3.45 GHz, 5.030 GHz to 5.091 GHz, 7.125 GHz to 8.400 GHz, and 18.1 GHz to 18.6 GHz bands toward DSA between commercial and DoD entities. AI-enabled perception calls for RF spectrum sensing and the detection of waveform parameters (e.g., modulation and perhaps even coding parameters [5]), carrier aggregation and bandwidth parts or similar information, propagation directions, power levels, and channel behaviors such as multipath, Doppler, and inter-symbol interference (ISI) metrics [6]. Typical spectrum sensors likely do not provide actionable intelligence on the spectrum except for usage statistics [7–10], where the detection of “spectrum holes” determines sub-band(s) possibly available for SUs [11–17]. Spectrum fraud is a major issue [18]. The presence of harmful RF interference (RFI) can only be proven by independent third-party sensing and perception. To internalize a model of the spectral landscape, we will need a good understanding of the measured RF environment; simple energy binning using an FFT is too limiting for future use cases. Mitigating RFI events everywhere all the time is challenging and calls for the detection of jamming, pulse weapons, and spectrum fraud. We need real-time perception using machine intelligence, as events happen everywhere all the time at rates of thousands of events per second, thus ruling out any human-in-the-loop approaches. The fully AI-enabled perception of RF environment is of critical importance and serves as an intelligent information source to drive automated DSA [19].

1.5. Contribution of the Paper

We propose a computationally efficient intelligent spectrum sensing system in this paper. We incorporate a DFT-based multibeam beamformer to directionally sense RF signals. In order to reduce the computational complexity of the multibeam beamformer, we employ approximate discrete Fourier transform (ADFT) instead of DFT. The digital processing back-end of the proposed multibeam beamformer is implemented on a field-programmable gate array (FPGA). The digital processing back-end connected to an antenna array consisting of 32 patch antennas operating at 5.7 GHz confirms the real-time spectrum sensing. Furthermore, the experimental setup demonstrates the application of ADFT-based multibeam beamforming in spectrum sensing, offering a promising approach for real-time, low-complexity, and efficient directional spectrum perception.

1.6. Organization of the Paper

The remainder of this paper is organized as follows. Section 2 provides a comprehensive review, beginning with an exploration of dynamic spectrum access, focusing on both signal processing techniques and machine learning/deep learning approaches, followed by a discussion on radio astronomy and radio frequency interference. Section 3 delves into the approximate discrete Fourier transform, offering insights into the theoretical background and the specific implementation of the ADFT. In Section 4, we present a detailed system overview, covering the proposed architecture, the analog front-end, the ADFT core and digital ADFT spectral estimation, digital FFT spectrometer, and digital high-speed connectivity. Section 5 outlines experimental results including calibration and beam measurements,

showcasing the practical performance of the proposed method. Finally, Section 6 discusses the implications of our findings and outlines potential directions for future research.

2. Review

We review related works in dynamic spectrum access in this section. We first consider signal processing techniques employed for spectrum sensing. We further consider the machine learning and recently proposed deep learning techniques in the first subsection. Next, we review radio astronomy with next-generation radio telescopes, in particular, the spectral sensing of RFIs.

2.1. Dynamic Spectrum Access: Signal Processing and ML/DL Approaches

DSA was first considered two decades ago based on the measurements of actual spectrum usage performed by the Spectrum Policy Task Force of Federal Communications Commission (FCC) [20]. On the contrary of the majorities' belief of a well-crowded spectrum, the actual measurements indicated that many portions of the spectrum are idle at a given time and location, leading to spectral holes. This results from the fixed spectrum access policy. Under DSA, secondary users (i.e., unlicensed users) are allowed to utilize spectral holes of primary users (i.e., licensed users) without interfering with the operation of primary users through a *cognitive radio* [21–24], which is an intelligent system built on a software-defined radio. However, there is an upper limit to the channel capacity that can be achieved with DSA [25]. Here, we briefly review *spectrum sensing* techniques associated with DSA, and the reader is referred to [26–28] for other aspects of DSA, such as spectrum management and spectrum sharing policies.

Several techniques have been proposed for both narrowband and wideband spectrum sensing. For narrowband sensing, the main non-ML techniques are energy detection [29–32], matched filter-based detection [33–35], cyclostationary feature detection [36,37], and covariance-based detection [38–40]. The energy detection and covariance-based detection approaches do not require prior knowledge of the primary user's signal; however, the former is not reliable at low signal-to-noise ratio (SNR) values. Matched filter-based detection provides optimal sensing and better detection at low SNR values, but this technique requires prior knowledge of the primary user's signal, which may not always be available. Cyclostationary feature detection is generally robust against noise uncertainty and works well at low SNR values; however, it suffers from a large sensing time. In wideband sensing, as a straightforward approach, a wideband signal may be divided into a set of narrowband signals via temporal DFT or a filter bank [41–44] and sense sequentially or in parallel using narrowband sensing techniques. However, sequential sensing requires more time, whereas parallel sensing requires more resources. Because a wideband signal with spectral can be considered a sparse signal in the frequency domain, several compressive sensing-based techniques have been proposed for wideband sensing [45–48]. These approaches require lower sampling rates and have lower power consumption in addition to the fast sensing. Thus, compressive sensing-based approaches eliminate most of the drawbacks of the sequential or parallel narrowband sensing of a wideband signal.

In the last decade, a considerable number of ML and DL techniques have been proposed for spectrum sensing. ML algorithms have been mostly used to determine the channel occupancy patterns of a primary user and to classify the channel as either free or occupied. In [49–53], K-means clustering and Gaussian mixture models have been first used to determine the presence of primary users, and a classifier such as a support vector machine and K-nearest-neighbor is then employed to determine whether a channel is free or occupied. DL techniques such as convolutional neural networks and long short-term memory have been employed for spectrum sensing in [54–57], which outperformed the conventional techniques. Recently, a transformer-based DL architecture was proposed in [58]. This DL approach outperforms the previously proposed approaches based on convolutional neural networks. Another DL approach employed for spectrum sensing in a cooperative way is deep reinforcement learning [59–62]. These deep reinforcement

learning approaches improve the robustness of the spectrum sensing system and allows it to make more accurate decisions in dynamic environments. The reader is referred to [63–65] for a comprehensive review of spectrum sensing techniques.

The spatio-temporal modeling of cognitive radio systems using multi-dimensional signal processing concepts was presented in [66]. Here, directional sensing was exploited using three-dimensional infinite impulse response (IIR) filters as beamformers. Similar directional sensing approaches using multi-dimensional filters were presented in [67–69]. Furthermore, antenna array systems were employed for directional sensing in [70–72]. In [73], a DFT-based multibeamformer was employed for real-time directional sensing. Compared to the multibeamformer employed with ADFT in this study, exact DFT was employed in [73] with 16 simultaneous beams at 2.4 GHz. We note that the upper limit of the channel capacity that can be achieved with DSA [25] is not changed using the ADFT instead of the DFT.

2.2. Radio Astronomy and RFI

Radio astronomy relies on a minimally contaminated radio spectrum to observe the universe [74]. Detecting and estimating weak celestial signals arriving from far away galaxies are particularly required in achieving the key science goals of the next-generation radio telescopes, i.e., the Square Kilometre Array (SKA) (<https://www.skao.int/> accessed on 24 July 2024) [75], the 2030 Atacama Large Millimeter/sub-millimeter Array—Wideband Sensitivity Upgrade (ALMA-WSU) (https://science.nrao.edu/facilities/alma/science_sustainability/wideband-sensitivity-upgrade accessed on 24 July 2024) [76], and the next-generation Very Large Array ngVLA (<https://ngvla.nrao.edu/> accessed on 24 July 2024) [77]. Despite being located at remote sparsely populated regions in the world, the increasing intensity, frequency, bandwidth, and occurrences of RFI are threatening the utility of the next-generation radio telescopes [78]. RFI from the global navigation satellite system (GNSS) [79], terrestrial and airborne wireless communications systems, satellite mega constellations [80], and other technologies is encroaching on the spectra that have been exclusively used by radio astronomers, making it harder to distinguish faint celestial signals from RFI.

The International Telecommunication Union (ITU) (<https://www.itu.int/en/Pages/default.aspx> accessed on 24 July 2024) and the FCC (<https://www.fcc.gov/> accessed on 24 July 2024) have important roles in protecting the spectrum for radio astronomy and other passive users. The ITU, through its radio communication sector (ITU-R), is responsible for developing international regulations and recommendations for the use of radio frequencies. It recognizes the significance of protecting the spectrum for passive services, such as radio astronomy, Earth exploration satellite services, and meteorological satellite services. The ITU-R identifies specific frequency bands for these passive services and establishes regulatory provisions to ensure their protection from harmful interference. These provisions include coordination procedures, power limits, and frequency separation requirements to safeguard the sensitive observations and measurements conducted by radio astronomers and other passive users [81,82]. Similarly, the FCC in the United States recognizes the importance of protecting the spectrum for radio astronomy and other passive services. The FCC's Office of Engineering and Technology (OET) formulates policies and rules to prevent harmful interference with these services. It establishes technical standards and licensing conditions that take into account the needs of passive users, including radio astronomers. Also, the FCC works closely with the National Science Foundation (NSF) and other relevant agencies to coordinate spectrum usage and protect the integrity of scientific observations. However, both organizations are under relentless pressure from the industry and governments for more commercial utilization of the spectrum that has been previously allocated for the non-commercial users.

Therefore, ideally, awareness of the key characteristics of RFI (i.e., frequency range, modulation scheme, source locations, intensity, etc.) would help radio telescope operators with planning and scheduling observations. Nevertheless, this is not practical for all

cases of RFI; therefore, records of radio spectral strength in terms of strength, direction, duration, and time of incident would also help the radio astronomers to flag and excise the contaminated observations [83]. In this case, the *omnipresent perception* of the EM environment would be a huge boon in the postprocessing stage for radio astronomical observations [84].

3. Approximate DFT

In this section, we review the fundamentals of DFT approximation by describing its technical rationale and mathematical formalism in terms of an optimization problem. Next, we focus on the state-of-the-art 32-point approximate DFT, which is numerically described along with its associate fast algorithm. An arithmetic complexity analysis for the discussed 32-point ADFT is also provided with a comparison with reference methods.

3.1. Theoretical Background

The discrete Fourier transform is a linear operator that relates an N -point input signal $\mathbf{x} = [x[0] \ x[1] \ \dots \ x[N-1]]^\top$ to the output signal $\mathbf{X} = [X[0] \ X[1] \ \dots \ X[N-1]]^\top$ according to the following expression:

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot \omega_N^{-nk}, \quad k = 0, 1, \dots, N-1, \quad (1)$$

where $\omega_N = \exp(2\pi j/N)$ is the N th root of the unity, and $j \triangleq \sqrt{-1}$. Computed by definition, the DFT presents a computational complexity in $O(N^2)$, which is prohibitively high for real-time applications.

In practice, the DFT is computed by means of one of many efficient numerical routines—fast algorithms, which are collectively referred to as “FFTs”. Particular popular choices of FFTs are the radix-2-based algorithms, such the Cooley–Tukey FFT, which can recursively decompose the N -point DFT block in two $N/2$ -point DFT sub-blocks [85].

In general, FFTs are capable of reducing the computational complexity of the DFT computation from $O(N^2)$ to $O(N \log N)$ [86]. In many cases [87–92], such a computational cost reduction achieves the theoretical minimum multiplicative complexity of the DFT [93]. As a consequence, it is impossible to further reduce the multiplicative complexity. This being a quite mature area of research, the proposition of new FFTs capable of offering significant reductions in computation cost is an unlikely event.

However, noticeable reductions can be accomplished under a fault-tolerant paradigm. If the DFT computation is permitted a given error level, then the computational effort can be re-adjusted to meet the lower, but acceptable, precision level. As a consequence, fewer arithmetical operations are needed for the DFT estimation [94].

3.2. Optimization Problem

Systematically, a way to obtain such low-complexity DFT estimators is by means of matrix approximation theory. In matrix formalism, the DFT is described by the following operation:

$$\mathbf{X} = \mathbf{F}_N \cdot \mathbf{x}, \quad (2)$$

where the N -point DFT matrix, denoted by $\mathbf{F}_N$, is the $N \times N$ matrix with elements given by $f_{k,n} = \omega_N^{-nk}$, $k, n = 0, 1, \dots, N-1$.

In the context of approximations for discrete transforms, qualitatively, an approximate discrete transform, $\hat{\mathbf{F}}_N$, is a transform matrix such that $\hat{\mathbf{X}} = \hat{\mathbf{F}}_N \cdot \mathbf{x} \approx \mathbf{X} = \mathbf{F}_N \cdot \mathbf{x}$. In other words, the exact and the approximate transform-domain signals are “close” in some relevant sense to be quantitatively defined. Generally, an approximation transform must satisfy the following conditions: (i) it can largely preserve meaningful properties of the exact transform; (ii) there is mathematical proximity between the exact and approximate transform matrix according to a contextually relevant metric function—usually a performance figure of

merit; and (iii) it possesses a significantly lower computational cost, compared to the exact computation by FFTs. In [95], an overview of the theory is made available.

Usually, in order to preserve the physical interpretation of the approximate spectrum, approximate transformation matrices are sought to be close in the Euclidean sense to the associate exact transformation matrix. This can be accomplished through the minimization of the Frobenius norm of the difference between the exact and candidate matrices for approximation, subject to the restriction of low complexity. Therefore, a low-complexity approximation can be obtained by restricting its elements to numerical sets of trivial multiplicands, such as $\{0, \pm 1\}$ or $\{0, \pm 1, \pm 2\}$.

In symbols, a possible formulation for the problem of deriving a DFT approximation is given by [95]

$$\min_{\hat{f}_{k,n}} \|\mathbf{F}_N - \mathbf{S} \cdot \hat{\mathbf{F}}_N\|_F \quad (3)$$

$$\text{subject to } \hat{f}_{k,n} \in \{0, \pm 1\}^2, \quad (4)$$

where $\hat{f}_{k,n}$, where $k, n = 0, 1, \dots, N-1$, denotes the entries of the approximate matrix $\hat{\mathbf{F}}_N$, $\|\cdot\|_F$ denotes the matrix Frobenius norm, and $\mathbf{S}$ is a normalization matrix used to ensure that the basis vectors of $\mathbf{S} \cdot \hat{\mathbf{F}}_N$ have energy equal or near to one.

3.3. 32-Point ADFT

In this work, we adopted the 32-point ADFT introduced mathematically in [96] and further elaborated in theory and in hardware in [97,98]. The explicit numerical form of the selected 32-point ADFT is given in Appendix A. Figure 1 shows a comparison of the magnitude responses for the 32-point exact DFT and the 32-point ADFT. The phase responses along the maxima—i.e., the beams—remain unchanged, and any deviation of phase only occurs in the deep stop band regions. Therefore, the influence of the approximation nature of the ADFT on the phases of the beamformed signals is negligible.

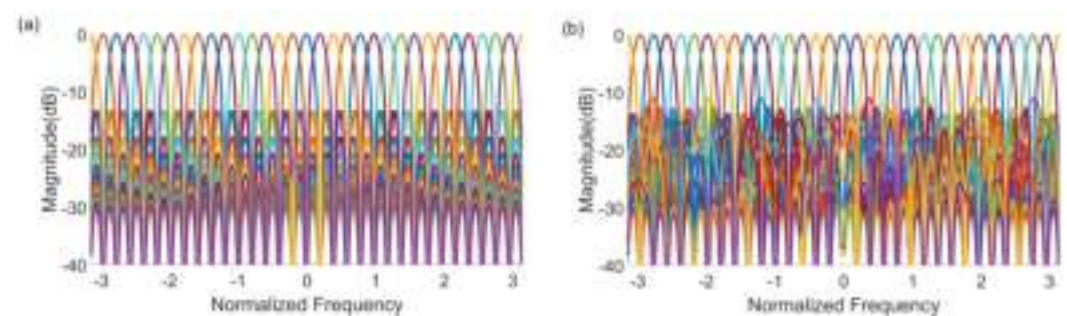


Figure 1. Simulated magnitude response of (a) ideal 32-point DFT and (b) low-complexity 32-point ADFT [97,98].

3.4. Fast Algorithm and Complexity Analysis

Represented by the low-complexity matrix $\hat{\mathbf{F}}_{32}$, the selected ADFT admits the following factorization:

$$\hat{\mathbf{F}}_{32} = \mathbf{W}_8 \cdot \mathbf{W}_7 \cdot \mathbf{W}_6 \cdot \mathbf{W}_5 \cdot \mathbf{W}_4 \cdot \mathbf{W}_3 \cdot \mathbf{W}_2 \cdot \mathbf{W}_1, \quad (5)$$

where $\mathbf{W}_i$, $i = 1, 2, \dots, 8$, are sparse, low-complexity matrices. In the Appendix A, we provide the explicit definition of matrices $\mathbf{W}_i$, $i = 1, 2, \dots, 8$.

For comparison, Table 1 lists the arithmetic operation count for the exact 32-point DFT computed according to its definition and to the radix-2 Cooley–Tukey FFT. For this operation counting, we assume that (i) the input signal is purely real-valued; (ii) trivial multiplications by 0, ± 1 , or $\pm j$ are not counted; and (iii) a multiplication between a real number and a complex number is performed as two real multiplications—not a complex multiplication [99]. Due to (i) and (iii), the 704 non-trivial multiplicands of $\hat{\mathbf{F}}_{32}$ are equivalent to 1408 real multiplications.

The evaluation of the number of additions considered the sum of the number of real- and imaginary-part additions. It was also provided the arithmetic complexity for the discussed 32-point ADFT by its definition and by means of its fast algorithm (factorization). The complexity analysis of the radix-2 Cooley–Tukey FFT is detailed in [99].

Table 1. Performance of the DFT, the Cooley–Tukey FFT, and the discussed 32-point ADFT. Trivial multiplications by ± 1 or $\pm j$ not counted; only real-valued input signals assumed.

Method	Real Multiplications	Real Additions
Exact 32-point DFT ($\mathbf{F}_{32}$)	1408	1666
Radix-2 Cooley–Tukey FFT ($N = 32$) [99]	88	408
Approximate DFT ($\hat{\mathbf{F}}_{32}$) [96]	0	1282
Fast algorithm for $\hat{\mathbf{F}}_{32}$ ($\prod_{i=1}^8 \mathbf{W}_i$) [97]	0	144

Table 2 shows the arithmetic complexity for each individual matrix term of the $\hat{\mathbf{F}}_{32}$ factorization. The matrix $\mathbf{W}_8$ does not present arithmetic costs because it consists of only sign-changing and data-swapping (real- and imaginary-part interchanging) operations. The data interchange is a consequence of the multiplication by j (a rotation by $\pi/2$).

Table 2. Performance of the matrix factors of the ADFT fast algorithm. Trivial multiplications by ± 1 or $\pm j$ not counted; only real-valued input signal assumed.

Operation	$\mathbf{W}_1$	$\mathbf{W}_2$	$\mathbf{W}_3$	$\mathbf{W}_4$	$\mathbf{W}_5$	$\mathbf{W}_6$	$\mathbf{W}_7$	$\mathbf{W}_8$
Real multiplications	0	0	0	0	0	0	0	0
Real additions	30	30	14	14	30	14	12	0

4. System Overview

In this section, we describe the design and realization of the antennas, microwave receivers, mixed-signal converter circuits, and FPGA-based digital realizations that constitute the proposed spatial–temporal spectrum perception system. The antennas and microwave systems were designed and realized in-house, while the FPGA system was procured from the open-source radio telescope instrumentation developer community (CASPER) that is in charge of major radio instruments, such as the Square Kilometer Array (SKA), Murchison Wide-field Array, the Allen Telescope Array, and others.

4.1. Proposed Architecture

To experimentally validate and measure the beams generated by the proposed ADFT algorithms, a 5.7 GHz RF system with a digital processing back-end was meticulously designed and implemented. Figure 2a shows the overall architecture of the system based on a 32-element antenna array, encompassing key subsystems such as the antenna array itself, the RF receiver chain, and the digital hardware unit. Figure 2b shows the experimental setup of the proposed architecture. In addition to the digital cores responsible for executing the spatial DFT of the sampled signals, supplementary circuitry was developed within the FPGA. This additional circuitry facilitates real-time data manipulation and beam measurements across different angles, enhancing the flexibility and capability of the system.

The re-configurable open architecture computing hardware (ROACH-2)-based digital signal processing (DSP) platform is shown in Figure 3, and it plays a crucial role by serving as an intermediary layer, enabling communication between the FPGA hardware (specifically its memory) and the onboard computer. The ROACH-2 board interfaces with the main host Linux server via a 1 Gbps Ethernet connection. A Python routine running on the host server is programmed to interact with the ROACH-2 platform, performing control functions and retrieving data from the FPGA memory while iteratively scanning through different angles. This unique system architecture not only ensures precise beam

measurements but also enhances the overall efficiency and automation of the experiment, providing a robust platform for validating advanced beamforming algorithms.

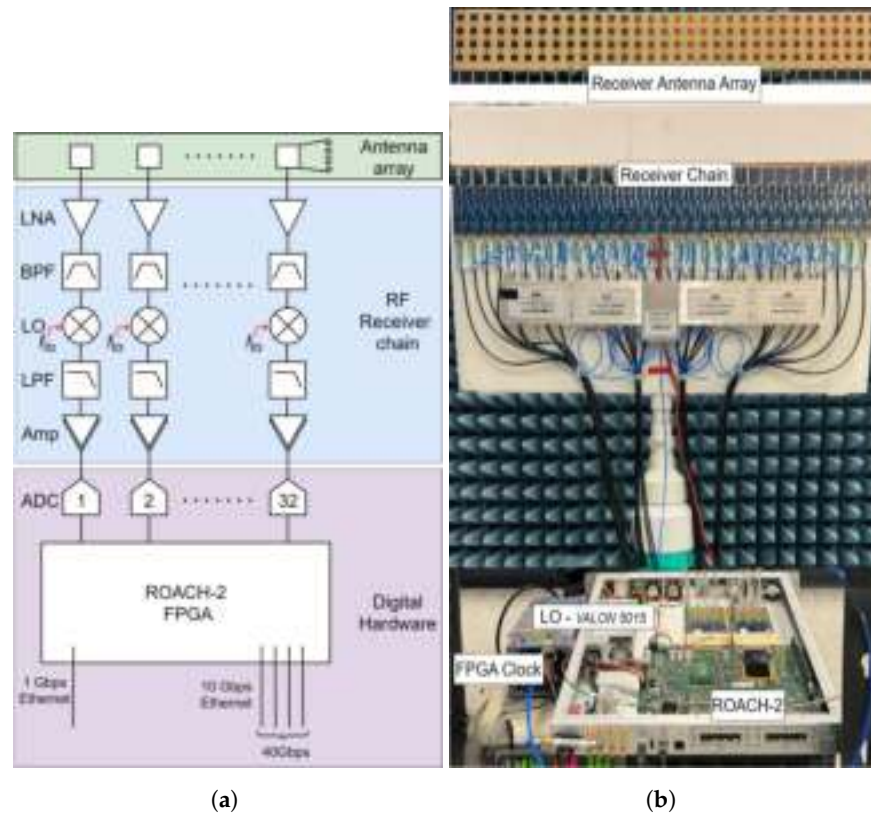


Figure 2. (a) The system architecture; (b) the experimental setup of the 32-beam array receiver operates at a 5.700 GHz to 5.800 GHz band followed by the ROACH-2 FPGA system (Xilinx Virtex-6 sx475T FPGA) digital back-end.



Figure 3. ROACH-2-based DSP platform based on Xilinx Virtex-6 Sx475 FPGA, and 32-channel ADC card. We gratefully acknowledge Dr. Dan Werthimer at UC Berkeley and the CASPER community for their contributions to the ROACH-2 and CASPER tools.

4.2. Analog Front-Ends

The RF front end of the receive-mode beamformer was constructed using a 32-element Uniform Linear Array (ULA) operating at 5.7 GHz, paired with a 32 direct conversion RF receiver chains on printed circuit boards (PCBs). The array elements are spaced at 0.6λ , which equates to approximately 32 mm at 5.7 GHz. Each antenna element is designed as a vertical subarray consisting of 4×1 patch antennas, facilitating passive beamforming in the vertical plane. This analog beamforming operates independently of the spatial FFT-based beamforming algorithms discussed in this work.

The outputs from the antenna elements are directly connected to 32 heterodyne receivers. These receivers utilize a centralized local oscillator (LO) system, which includes a low-phase-noise oscillator linked to a 32-output power divider network. Each receiver's initial stage features a low-noise amplifier (LNA) that delivers 16 dB gain at 5.8 GHz with a noise figure of 2.4 dB. Following amplification, the signals are filtered through a bandpass filter spanning 4.7 GHz to 6 GHz to eliminate out-of-band noise and interference. The filtered signals are then mixed and passed through low-pass filters to produce downconverted low intermediate frequency (IF) inputs.

Subsequently, these 32 low-IF signals are amplified by approximately 30 dB and digitized using two ADC16x250-8 analog-to-digital converter (ADC) cards. Each ADC card supports 16 single-ended input channels, provides 8-bit resolution, and operates at sampling rates up to 250 Msample/s per channel. This setup ensures the high-fidelity digitization of the received signals for further processing [97].

4.3. ADFT Cores and Digital ADFT Spectral Estimation

The discussed 32-point ADFT was implemented by utilizing Xilinx tools. The sparse matrices from the factorization shown in (5) were structured as parallel input–output blocks to optimize performance [97,98]. The verification of these blocks was conducted through MATLAB Simulink 8.5, ensuring the reliability of the design before hardware deployment. Subsequently, the verified ADFT design was mapped onto the Xilinx Virtex-6 sx475T FPGA.

To integrate the ADFT core into the ROACH-2 FPGA platform, we utilized two libraries from the Collaboration for Astronomy Signal Processing and Electronics Research (CASPER) community [100,101]. This integration enabled the system to generate 32 simultaneous RF beams in parallel. Each beam corresponds to a discrete direction, calculated using the function $\theta_k = \arcsin(k/32)$, where k ranges from -16 to 15 . The successful implementation of the ADFT on the ROACH-2 FPGA not only validated the algorithm effectiveness but also demonstrated the practical application of advanced digital hardware design techniques. This work highlights the potential of ADFT in various applications requiring efficient RF beamforming.

We implemented a spatial windowing technique to address the issue of high side lobes in the ADFT, where the worst-case side lobe was observed to be 1.5 dB higher than that of the (exact) DFT. Four configurations were tested: three windowing functions (Butterworth, Hamming, and Hanning) and a baseline with no windowing [85]. The selection among these windowing functions was managed using a multiplexer equipped with a control selector pin.

Figure 4 shows the spatial windowing unit coupled with the ADFT sparse matrices. In mathematical terms, the windowing is introduced as a diagonal matrix. The ADFT-based spectral estimation can be described by combining the windowing diagonal matrix ($\mathbf{D}_{\text{windowing}}$), the low-complexity ADFT core ($\hat{\mathbf{F}}_{32}$), and a power-normalizing diagonal matrix ($\mathbf{D}_{\text{normalizer}}$). The mathematical representation of the proposed approximate spectrum estimation is given by

$$\text{ADFT}_{\text{estimator}} = \mathbf{D}_{\text{normalizer}} \cdot \hat{\mathbf{F}}_{32} \cdot \mathbf{D}_{\text{windowing}}. \quad (6)$$

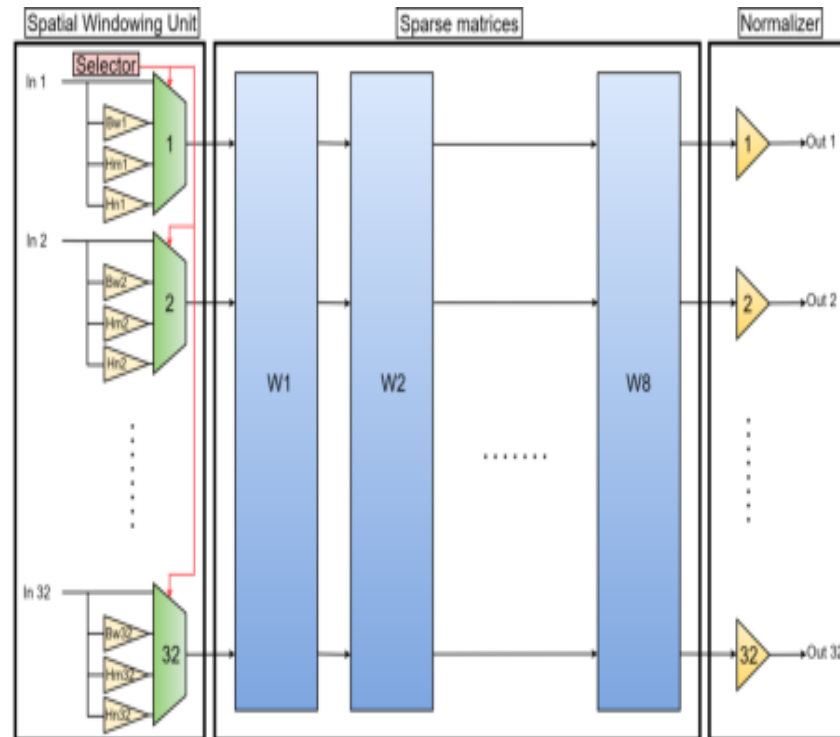


Figure 4. ADFT architecture with spatial windowing and power normalizing units.

Figure 5 illustrates the ADFT response and its modifications under the windowing techniques. It is noteworthy that the introduction of windowing functions resulted in wider main lobes, which is a compromise made to suppress the side lobes. This trade-off between main lobe width and side lobe suppression is a consideration in the application of spatial windowing techniques. However, implementing spatial windowing increases the hardware complexity due to the multiplexers introduced by the windowing unit. Therefore, the user must select the appropriate configuration based on their specific requirements and constraints.

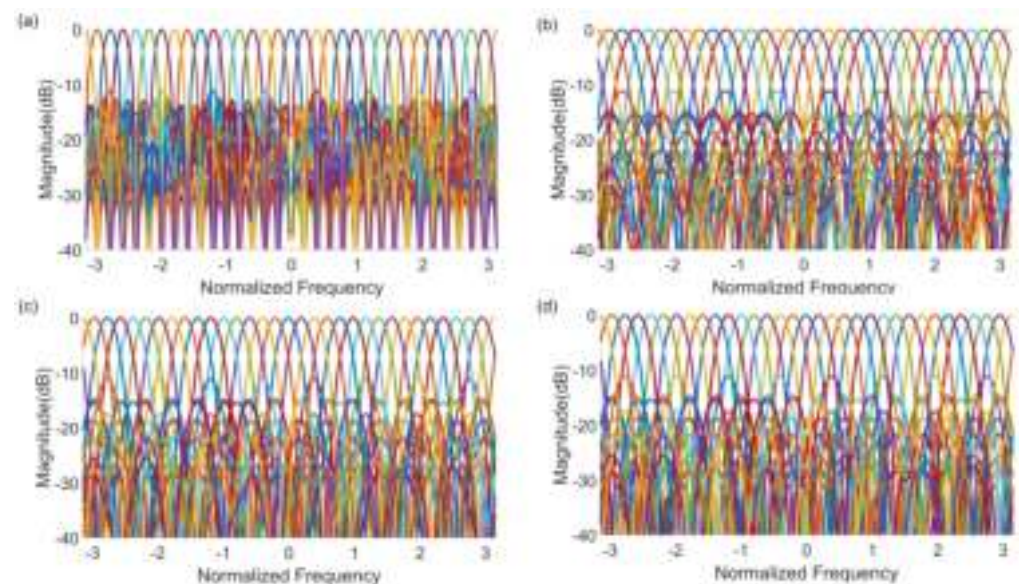


Figure 5. Simulated magnitude response of 32-point ADFT and its modifications under three different windowing techniques. Subplot (a) shows the baseline response without any windowing. Subplots (b–d) demonstrate the ADFT responses after applying the Butterworth, Humming, and Hanning windows, respectively.

4.4. Digital FFT Spectrometers

The system described involves the implementation of a spatial FFT that generates 32 parallel outputs, each corresponding to a distinct spatial RF beam. Subsequent to this spatial FFT, a temporal FFT spectrometer is applied to each of these 32 spatial RF beams, achieving channelization in the temporal frequency domain. Each temporal FFT is computed with 1024 points, necessitating the use of 32 parallel FFT cores. This parallel processing is facilitated by leveraging real-sampled biplex FFT blocks available in the Xilinx Virtex-6 SX475T FPGA.

The real-sampled biplex FFT blocks exploit the Hermitian conjugation method, enabling the computation of the real-sampled FFT. This technique utilizes a complex FFT core to simultaneously transform two real-valued data streams. As a result, the FFT operation produces all frequencies, both positive and negative, with negative frequencies being the mirror images of their positive counterparts [100].

The temporal FFT outputs for each beam yield 1024 complex values corresponding to 1024 discrete frequency bins. The power spectral density (PSD) is estimated by squaring the magnitude of each frequency bin. This estimate is derived by averaging across 1024 temporal FFT evaluations, resulting in a new PSD measurement every 1 million clock cycles. A block random access memory (BRAM) vector accumulator block collects the PSD values and transfers them to the software process supported by the CASPER design flow. Figure 6 shows the overall ADFT and temporal FFT flow, including energy integration and BRAM registers. The casperfpga Python library is used to interact with CASPER hardware, extracting values from the shared BRAM.

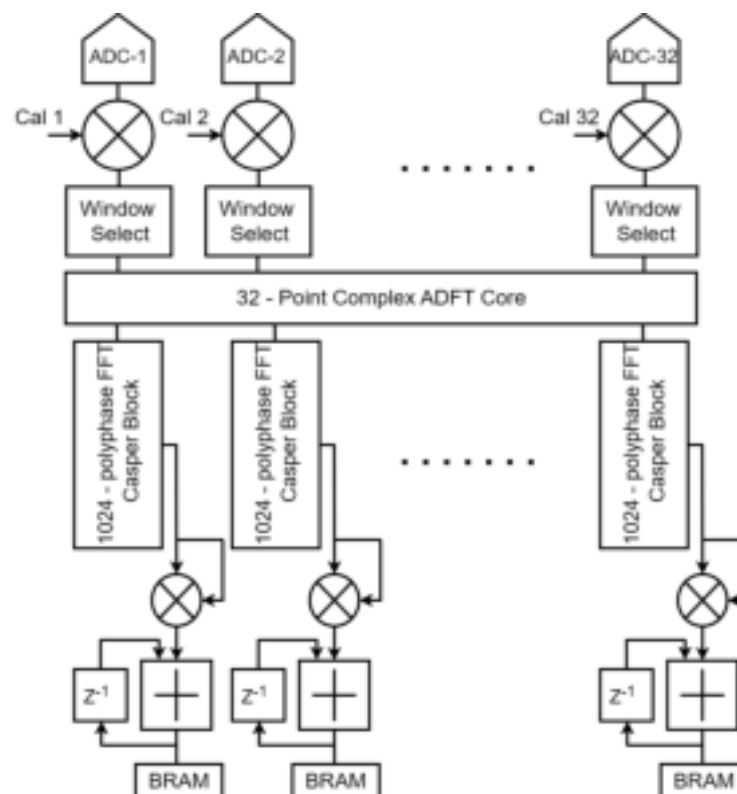


Figure 6. ADFT and FFT calibration, energy integration, and overall system block including BRAM registers.

The FPGA operates with a clock frequency set to 200 MHz. Given this configuration, the system frequency resolution for each beam is determined to be $\Delta F = 100/1024$ MHz, which is approximately 100 kHz. The Nyquist bandwidth of the system is thus 100 MHz. This configuration ensures precise channelization and high-resolution frequency analysis for each of the 32 spatial RF beams.

4.5. Digital High-Speed Connectivity

The ROACH-2 boards are equipped with four SFP+ ports on a single 10 GbE mezzanine card, where the Ethernet interface is driven by an on-board 156.25 MHz crystal oscillator. This clock frequency is internally multiplied by a factor of 66 within the FPGA, resulting in an effective wire speed of 10.3125 Gbps.

The CASPER 10 GbE Simulink core facilitates the transmission and reception of UDP packets over IPv4, encapsulated within Ethernet frames. Each Ethernet frame incurs an overhead of 38 bytes for the frame header, 20 bytes for the IPv4 header, and 16 bytes for the UDP header, totaling at least 74 bytes of overhead per packet. To align with 64-bit boundaries, some headers may also include zero-padding, further impacting the effective throughput. The maximum payload size for the CASPER 10 GbE core is 8192 bytes, implemented in the BRAM, with an additional 512 bytes available in distributed RAM for application-specific headers [100].

5. Experimental Results

5.1. Calibration

The calibration of the RF array system is essential for optimal beamforming performance and was carried out in two distinct phases. Initially, the calibration focused on the ADCs, utilizing open-source routines previously developed by CASPER group members for this hardware. The second phase addressed the digital correction of mismatches in the microwave front end. Relative gain and phase mismatches of the IF outputs for each chain were determined in relation to a reference chain using a 5.71 GHz reference carrier input. The relative gain differences were compensated using digital multipliers, and gain scale factors were calculated by the measuring average power values of each ADC output. Ideally, for phase difference compensation, fractional delay filters are required, as the RF chains could have smaller delays than the sampling period value. As an interim solution, integer delay filters were designed and implemented in the hardware to mitigate phase differences.

5.2. Beam Measurements

Power patterns were obtained by transmitting a continuous-wave (CW) signal at $f_{\text{RF}} = 5.71$ GHz. The LO signal frequency f_{LO} determines the IF as $f_{\text{RF}} - f_{\text{LO}}$. For the measurements in this study, f_{LO} was set to 5.7 GHz, resulting in an IF of 10 MHz, with the downconverted outputs digitized at $f_{\text{CLK}} = 200$ MHz.

Digital integrators at each ADFT bin output were used to compute the received energy over a fixed duration. Spatial beamforming occurs at a rate of 200 MHz, and the PSD is calculated in real time by squaring the magnitude of each frequency bin. This estimate is derived by averaging over 1024 temporal FFT computations performed at 200 MHz. The geometry of the measurement setup significantly influences the measured array factor of the beams. Ideally, the transmitter and receiver should be sufficiently far apart so that the waves incident on the receiver array can be approximated as plane waves. According to standard guidelines [102], the separation between the transmitter and receiver should exceed 20 m at 5.7 GHz to ensure that the receiver aperture is in the far field. The beams were measured on an open parking deck with a transmitter–receiver separation of approximately 20 m.

Figure 7 illustrates the measured array factor patterns for beams 0 to 16. Due to the input of real-valued IF signals into the ADFT, the resulting beam patterns exhibit symmetry, producing mirror images for the remaining bins. Additionally, the presence of two mirror peaks is observed within bins 0 to 16. The measured beams conform closely to the expected patterns, maintaining a separation of more than 10 dB from the other beam. Each ADFT output bin corresponds to an RF far-field beam directed at a discrete angle given by $\theta_k = \arcsin(k/N)$, where $k = -16, -15, \dots, 14, 15$, and $N = 32$ (number of array elements). This configuration results in a beam separation of $180^\circ/32 = 5.625^\circ$, providing an angular resolution of $\pm 2.8125^\circ$.

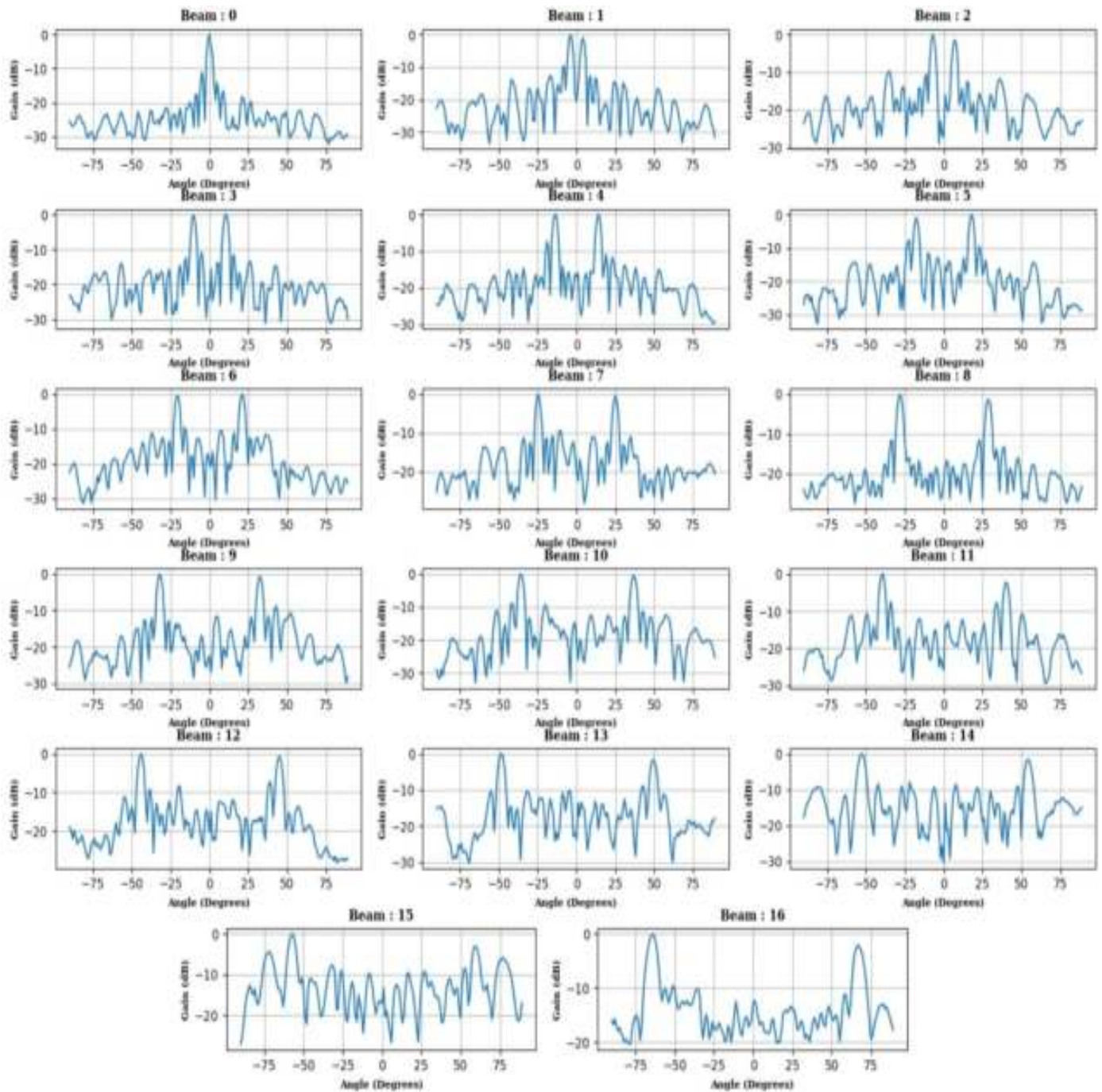


Figure 7. Sixteen beams measured from the 5.7GHz array (vertical axis is in decibel, and the horizontal axis is the azimuthal angle $[-90^\circ, 90^\circ]$). Each subfigure contains the measured array factor patterns of the beam using the real inputs of the ADFT core. The imaginary component of each ADFT input is set to zero here, thus resulting in symmetrical main lobes as expected.

Figure 8 depicts the measured temporal PSD distribution of respective beams at multiple frequencies when the signal source is positioned at a broadside angle of 15° , 35° , and 50° . The corresponding experimental setup with relative source locations and RF frequencies being emitted for testing purposes is shown in Figure 9.

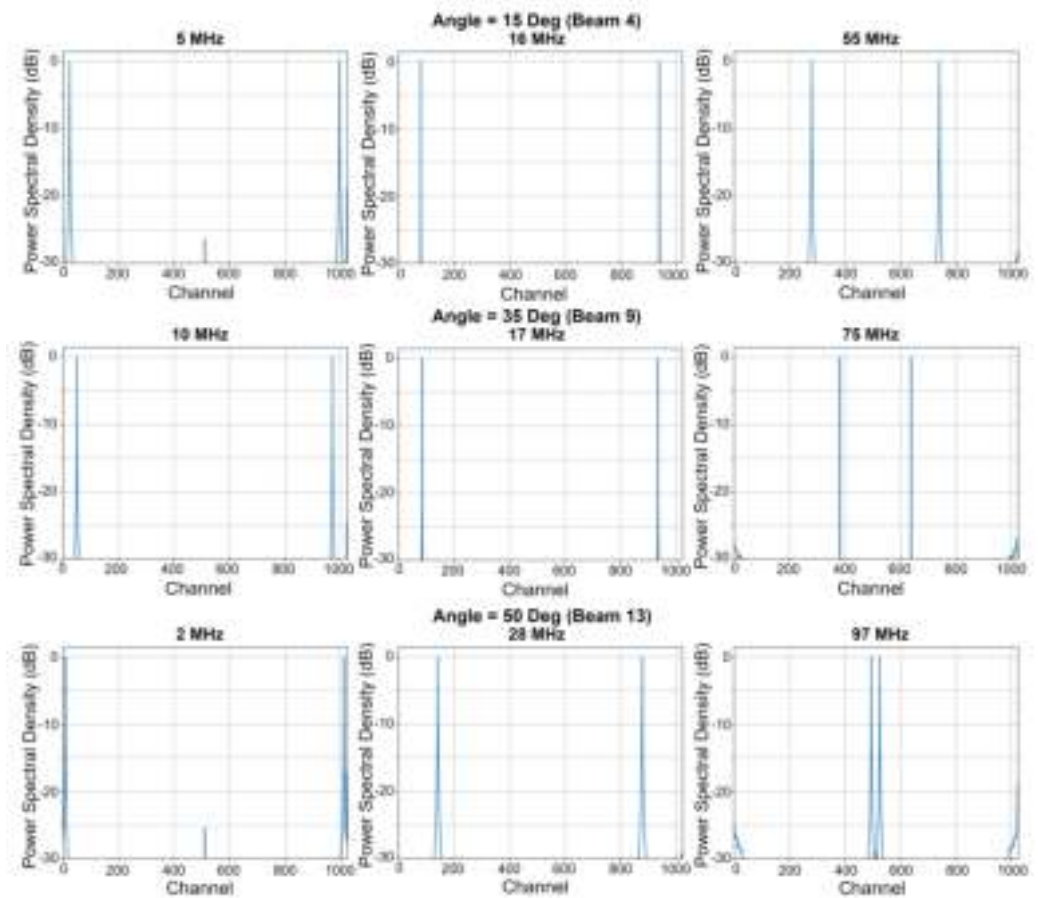


Figure 8. Temporal PSD over the 1024 discrete frequency bins for the particular RF beam at multiple frequencies when the source is placed at 15°, 35°, and 50° broadside receiver angles.

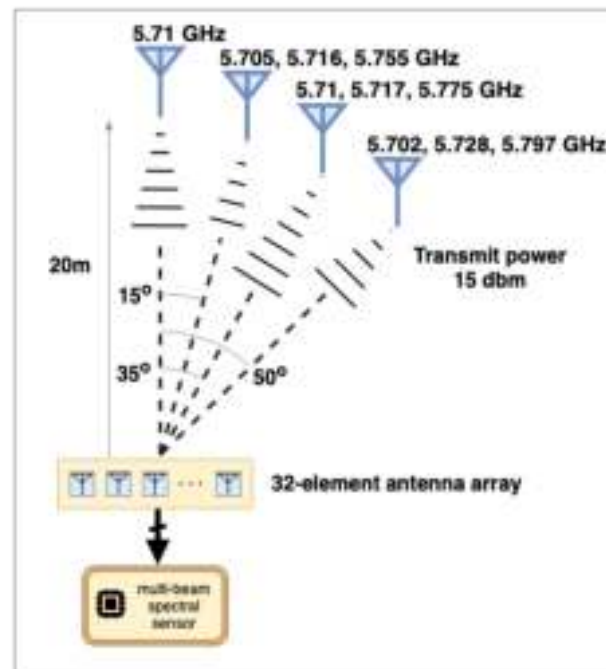


Figure 9. Illustration of the overall setup showing the transmitting antennas, receiving antenna array and the multibeam spectral sensor. The 32-element antenna array receiver captures signals from different directions, with the transmitters located 20 m away from the receiver.

6. Discussion

This paper presents a low-complexity digital beamforming architecture designed to generate 32 simultaneous RF beams using a 32-point DFT approximation, effectively eliminating multiplication operations. We validated the proposed method on a fully operational 32-element digital 1-D receiver array operating at 5.7 GHz. The 1-D array employs 32 parallel ADCs for sampling the antenna outputs and uses an ADFT implemented on a Xilinx FPGA to compute 32 RF beams in real time. Additionally, this paper discusses the enhancement of our existing 32-channel 5.7 GHz spectrum sensing system through the implementation of 32 parallel 1024-bin temporal FFTs on a ROACH-2 hardware platform. This experimental verification with real-time measurements across a band of frequencies and angles of incidence demonstrates the capability of the architecture to sense information from multiple directions with a frequency resolution of 100 kHz and directional resolution of about 6° .

The authors plan on expanding the ADFT to larger numbers of elements and create multi-dimensional ADFT beamformers, for beam-steering in both elevation and azimuthal planes. For example, using a 32×8 array, we could expand the set of beams to a rectangular grid of size 32×8 where both 32-point and 8-point FFTs are replaced by the corresponding ADFT algorithms based on sparse matrix factorization [103–106].

Author Contributions: Conceptualization and system design by A.M. Theory and design of ADFT and fast algorithm by R.J.C. Multi-dimensional signal processing review by C.U.S.E. Review of radio astronomy interference issues by T.G. Digital implementation of ADFT core by K.L. Experimental setup, digital design of CASPER tools, calibration, and real-time experimentation by S.S. and B.U.K. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by NSF (under the grants FuSE 2329012, SWIFT 2229471, ICORPS 2243346, SpectrumX), NTIA, CNPq (Brazil) under the grant 315047/23-2 and Senate Research Committee, University of Mortuwa under the grant SRC/LT/2020/08.

Data Availability Statement: All original contributions from this study are documented within the article. For further details, please contact the corresponding authors.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ADC	Analog-to-digital converter;
ADFT	Approximate discrete Fourier transform;
AI	Artificial intelligence;
ALMA	Atacama large millimeter array;
BRAM	Block random access memory;
CASPER	Collaboration for astronomy signal processing and electronics research;
CSI	Channel state information;
CW	Continuous wave;
DFT	Discrete Fourier transform;
DL	Deep learning;
DSA	Dynamic spectrum access;
DSP	Digital signal processing;
FCC	Federal communication commission;
FFT	Fast Fourier transform;
FPGA	Field-programmable gate array;
GNSS	Global navigation satellite system;
IF	Intermediate frequency;

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

and

and

1. Tolkien, J. *The Hobbit & The Lord of the Rings*; Houghton Mifflin Harcourt: Boston, MA, USA, 1954.
2. Kang, S.; Mezzavilla, M.; Rangan, S.; Madanayake, A.; Venkatakrishnan, S.B.; Hellbourn, G.; Ghosh, M.; Rahmani, H.; Dhananjay, A. Cellular Wireless Networks in the Upper Mid-Band. *IEEE Open J. Commun. Soc.* **2024**, *5*, 2058–2075. [[CrossRef](#)]
3. The White House. *National Spectrum Strategy*; The White House: Washington, DC, USA, 2023.
4. Restuccia, F.; Melodia, T. Deep Learning at the Physical Layer: System Challenges and Applications to 5G and Beyond. *IEEE Commun. Mag.* **2020**, *58*, 58–64. [[CrossRef](#)]
5. Triaridis, K.; Doumanidis, C.; Chatzidiamentis, N.D.; Karagiannidis, G.K. MM-Net: A Multi-Modal Approach towards Automatic Modulation Classification. *IEEE Commun. Lett.* **2023**, *28*, 328–331. [[CrossRef](#)]
6. Baldesi, L.; Restuccia, F.; Melodia, T. ChARM: NextG Spectrum Sharing through Data-Driven Real-Time O-RAN Dynamic Control. In Proceedings of the IEEE International Conference on Computer Communications (INFOCOM), London, UK, 2–5 May 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 240–249.
7. Zhang, L.; Xiao, M.; Wu, G.; Alam, M.; Liang, Y.C.; Li, S. A Survey of Advanced Techniques for Spectrum Sharing in 5G Networks. *IEEE Wirel. Commun.* **2017**, *24*, 44–51. [[CrossRef](#)]
8. Hu, F.; Chen, B.; Zhu, K. Full Spectrum Sharing in Cognitive Radio Networks toward 5G: A Survey. *IEEE Access* **2018**, *6*, 15754–15776. [[CrossRef](#)]
9. Shokri-Ghadikolaei, H.; Boccardi, F.; Fischione, C.; Fodor, G.; Zorzi, M. Spectrum Sharing in mmWave Cellular Networks via Cell Association, Coordination, and Beamforming. *IEEE J. Sel. Areas Commun.* **2016**, *34*, 2902–2917. [[CrossRef](#)]
10. Lv, L.; Chen, J.; Ni, Q.; Ding, Z.; Jiang, H. Cognitive Non-Orthogonal Multiple Access with Cooperative Relaying: A New Wireless Frontier for 5G Spectrum Sharing. *IEEE Commun. Mag.* **2018**, *56*, 188–195. [[CrossRef](#)]
11. Agarwal, S.; De, S. eDSA: Energy-efficient Dynamic Spectrum Access Protocols for Cognitive Radio Networks. *IEEE Trans. Mob. Comput.* **2016**, *15*, 3057–3071. [[CrossRef](#)]

12. Chiwewe, T.M.; Hancke, G.P. Fast Convergence Cooperative Dynamic Spectrum Access for Cognitive Radio Networks. *IEEE Trans. Ind. Inform.* **2017**, *14*, 3386–3394. [CrossRef]
13. Chang, H.H.; Song, H.; Yi, Y.; Zhang, J.; He, H.; Liu, L. Distributive Dynamic Spectrum Access through Deep Reinforcement Learning: A Reservoir Computing-based Approach. *IEEE Internet Things J.* **2018**, *6*, 1938–1948. [CrossRef]
14. Nguyen, H.Q.; Nguyen, B.T.; Dong, T.Q.; Ngo, D.T.; Nguyen, T.A. Deep Q-Learning with Multiband Sensing for Dynamic Spectrum Access. In Proceedings of the IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN), Seoul, Republic of Korea, 22–25 October 2018; pp. 1–5. [CrossRef]
15. Naparstek, O.; Cohen, K. Deep Multi-user Reinforcement Learning for Distributed Dynamic Spectrum Access. *IEEE Trans. Wirel. Commun.* **2019**, *18*, 310–323. [CrossRef]
16. Wang, S.; Liu, H.; Gomes, P.H.; Krishnamachari, B. Deep Reinforcement Learning for Dynamic Multichannel Access in Wireless Networks. *IEEE Trans. Cogn. Commun. Netw.* **2018**, *4*, 257–265. [CrossRef]
17. Yu, Y.; Wang, T.; Liew, S.C. Deep-Reinforcement Learning Multiple Access for Heterogeneous Wireless Networks. *IEEE J. Sel. Areas Commun.* **2019**, *37*, 1277–1290. [CrossRef]
18. Reston, V.A. Ligado Networks Sues U.S. Government for Unlawful and Uncompensated Taking of Ligado’s Exclusive FCC-Licensed Spectrum. 2023. Available online: <https://ligado.com/press/ligado-networks-sues-u-s-government-for-unlawful-and-uncompensated-taking-of-ligados-exclusive-fcc-licensed-spectrum/> (accessed on 28 May 2024).
19. Barden, S.; Dewdney, P.; Friesen, R.; Murowinski, R.; Savanandam, S. *ATD Strategic R&D Review Report—Signal Processing Portfolio*; Herzberg Astronomy and Astrophysics Research Centre-National Research Council Canada: Victoria, BC, Canada, 2022.
20. FCC Spectrum Policy Task Force. *Report of the Spectrum Efficiency Working Group*, Nov. 2002; FCC: Washington, DC, USA, 2024.
21. Mitola, J.; Maguire, G. Cognitive radio: Making software radios more personal. *IEEE Pers. Commun.* **1999**, *6*, 13–18. [CrossRef]
22. Haykin, S. Cognitive radio: Brain-empowered wireless communications. *IEEE J. Sel. Areas Commun.* **2005**, *23*, 201–220. [CrossRef]
23. Liang, Y.C.; Chen, K.C.; Li, G.Y.; Mahonen, P. Cognitive radio networking and communications: An overview. *IEEE Trans. Veh. Technol.* **2011**, *60*, 3386–3407. [CrossRef]
24. Wang, B.; Liu, K.R. Advances in cognitive radio networks: A survey. *IEEE J. Sel. Top. Signal Process.* **2011**, *5*, 5–23. [CrossRef]
25. Talebi, S.P. Primary service outage and secondary service performance in cognitive radio networks. *Wirel. Commun. Mob. Comput.* **2015**, *15*, 1982–1990. [CrossRef]
26. Zhao, Q.; Sadler, B.M. A Survey of Dynamic Spectrum Access. *IEEE Signal Process. Mag.* **2007**, *24*, 79–89. [CrossRef]
27. Ji, Z.; Liu, K.R. Dynamic Spectrum Sharing: A Game Theoretical Overview. *IEEE Commun. Mag.* **2007**, *45*, 88–94. [CrossRef]
28. Bhattarai, S.; Park, J.M.J.; Gao, B.; Bian, K.; Lehr, W. An Overview of Dynamic Spectrum Sharing: Ongoing Initiatives, Challenges, and a Roadmap for Future Research. *IEEE Trans. Cogn. Commun. Netw.* **2016**, *2*, 110–128. [CrossRef]
29. Zeng, Y.; Liang, Y.C.; Zhang, R. Blindly Combined Energy Detection for Spectrum Sensing in Cognitive Radio. *IEEE Signal Process. Lett.* **2008**, *15*, 649–652. [CrossRef]
30. Ling, X.; Wu, B.; Wen, H.; Ho, P.H.; Bao, Z.; Pan, L. Adaptive Threshold Control for Energy Detection Based Spectrum Sensing in Cognitive Radios. *IEEE Wirel. Commun. Lett.* **2012**, *1*, 448–451. [CrossRef]
31. Ranjan, A.; Anurag, Singh, B. Design and analysis of spectrum sensing in cognitive radio based on energy detection. In Proceedings of the International Conference on Signal and Information Processing, Nanded, India, 6–8 October 2016; pp. 1–5.
32. Arjoune, Y.; Mrabet, Z.E.; Ghazi, H.E.; Tamtaoui, A. Spectrum sensing: Enhanced energy detection technique based on noise measurement. In Proceedings of the IEEE Annual Computing and Communication Workshop and Conference, Las Vegas, NV, USA, 8–10 January 2018; pp. 828–834.
33. Jiang, C.; Li, Y.; Bai, W.; Yang, Y.; Hu, J. Statistical matched filter based robust spectrum sensing in noise uncertainty environment. In Proceedings of the International Conference on Communication Technology, Chengdu, China, 9–11 November 2012; pp. 1209–1213.
34. Zhang, X.; Chai, R.; Gao, F. Matched filter based spectrum sensing and power level detection for cognitive radio network. In Proceedings of the IEEE Global Conference on Signal and Information Processing, Atlanta, GA, USA, 3–5 December 2014; pp. 1267–1270.
35. Lv, Q.; Gao, F. Matched filter based spectrum sensing and power level recognition with multiple antennas. In Proceedings of the IEEE China Summit and International Conference on Signal and Information Processing, Chengdu, China, 12–15 July 2015; pp. 305–309.
36. Ilyas, I.; Paul, S.; Rahman, A.; Kundu, R.K. Comparative evaluation of cyclostationary detection based cognitive spectrum sensing. In Proceedings of the IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference, New York, NY, USA, 20–22 October 2016; pp. 1–7.
37. Cohen, D.; Eldar, Y.C. Compressed cyclostationary detection for Cognitive Radio. In Proceedings of the 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), New Orleans, LA, USA, 5–9 March 2017; pp. 3509–3513.
38. Zeng, Y.; Liang, Y.C. Spectrum-Sensing Algorithms for Cognitive Radio Based on Statistical Covariances. *IEEE Trans. Veh. Technol.* **2009**, *58*, 1804–1815. [CrossRef]
39. Zeng, Y.; Liang, Y.C. Eigenvalue-based spectrum sensing algorithms for cognitive radio. *IEEE Trans. Commun.* **2009**, *57*, 1784–1793. [CrossRef]
40. Zayen, B.; Hayar, A.; Kansanen, K. Blind Spectrum Sensing for Cognitive Radio Based on Signal Space Dimension Estimation. In Proceedings of the 2009 IEEE International Conference on Communications, Dresden, Germany, 14–18 June 2009; pp. 1–5.

41. Farhang-Boroujeny, B. Filter Bank Spectrum Sensing for Cognitive Radios. *IEEE Trans. Signal Process.* **2008**, *56*, 1801–1811. [\[CrossRef\]](#)
42. Quan, Z.; Cui, S.; Sayed, A.H.; Poor, H.V. Optimal Multiband Joint Detection for Spectrum Sensing in Cognitive Radio Networks. *IEEE Trans. Signal Process.* **2009**, *57*, 1128–1140. [\[CrossRef\]](#)
43. Raghu, I.; Chowdary, S.S.; Elias, E. Efficient spectrum sensing for Cognitive Radio using Cosine Modulated Filter Banks. In Proceedings of the IEEE Region 10 Conference, Singapore, 22–25 November 2016; pp. 2086–2089.
44. Kumar, A.; Saha, S.; Bhattacharya, R. Improved wavelet transform based edge detection for wide band spectrum sensing in Cognitive Radio. In Proceedings of the USNC-URSI Radio Science Meeting, Fajardo, Puerto Rico, 26 June–1 July 2016; pp. 21–22.
45. Polo, Y.L.; Wang, Y.; Pandharipande, A.; Leus, G. Compressive wide-band spectrum sensing. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, Taipei, Taiwan, 19–24 April 2009; pp. 2337–2340.
46. Arjoune, Y.; Kaabouch, N. Wideband Spectrum Sensing: A Bayesian Compressive Sensing Approach. *Sensors* **2018**, *18*, 1839. [\[CrossRef\]](#) [\[PubMed\]](#)
47. Li, Z.; Chang, B.; Wang, S.; Liu, A.; Zeng, F.; Luo, G. Dynamic Compressive Wide-Band Spectrum Sensing Based on Channel Energy Reconstruction in Cognitive Internet of Things. *IEEE Trans. Ind. Inform.* **2018**, *14*, 2598–2607. [\[CrossRef\]](#)
48. Yang, L.; Fang, J.; Duan, H.; Li, H. Fast Compressed Power Spectrum Estimation: Toward a Practical Solution for Wideband Spectrum Sensing. *IEEE Trans. Wirel. Commun.* **2020**, *19*, 520–532. [\[CrossRef\]](#)
49. Ding, G.; Wu, Q.; Yao, Y.D.; Wang, J.; Chen, Y. Kernel-Based Learning for Statistical Signal Processing in Cognitive Radio Networks: Theoretical Foundations, Example Applications, and Future Directions. *IEEE Signal Process. Mag.* **2013**, *30*, 126–136. [\[CrossRef\]](#)
50. Thilina, K.M.; Choi, K.W.; Saquib, N.; Hossain, E. Machine Learning Techniques for Cooperative Spectrum Sensing in Cognitive Radio Networks. *IEEE J. Sel. Areas Commun.* **2013**, *31*, 2209–2221. [\[CrossRef\]](#)
51. Mikaeil, A.M.; Guo, B.; Wang, Z. Machine Learning to Data Fusion Approach for Cooperative Spectrum Sensing. In Proceedings of the International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, Shanghai, China, 13–15 October 2014; pp. 429–434.
52. Ghazizadeh, E.; Nikpour, B.; Moghadam, D.A.; Nezamabadi-pour, H. A PSO-based weighting method to enhance machine learning techniques for cooperative spectrum sensing in CR networks. In Proceedings of the Conference on Swarm Intelligence and Evolutionary Computation, Bam, Iran, 9–11 March 2016; pp. 113–118.
53. Khalfi, B.; Zaid, A.; Hamdaoui, B. When machine learning meets compressive sampling for wideband spectrum sensing. In Proceedings of the International Wireless Communications and Mobile Computing Conference, Valencia, Spain, 26–30 June 2017; pp. 1120–1125.
54. Lee, W.; Kim, M.; Cho, D.H. Deep Cooperative Sensing: Cooperative Spectrum Sensing Based on Convolutional Neural Networks. *IEEE Trans. Veh. Technol.* **2019**, *68*, 3005–3009. [\[CrossRef\]](#)
55. Liu, C.; Wang, J.; Liu, X.; Liang, Y.C. Deep CM-CNN for Spectrum Sensing in Cognitive Radio. *IEEE J. Sel. Areas Commun.* **2019**, *37*, 2306–2321. [\[CrossRef\]](#)
56. Soni, B.; Patel, D.K.; López-Benítez, M. Long Short-Term Memory Based Spectrum Sensing Scheme for Cognitive Radio Using Primary Activity Statistics. *IEEE Access* **2020**, *8*, 97437–97451. [\[CrossRef\]](#)
57. Peng, Q.; Gilman, A.; Vasconcelos, N.; Cosman, P.C.; Milstein, L.B. Robust Deep Sensing through Transfer Learning in Cognitive Radio. *IEEE Wirel. Commun. Lett.* **2020**, *9*, 38–41. [\[CrossRef\]](#)
58. Zhang, W.; Wang, Y.; Chen, X.; Cai, Z.; Tian, Z. Spectrum Transformer: An Attention-Based Wideband Spectrum Detector. *IEEE Trans. Wirel. Commun.* **2024**, 1–1. [\[CrossRef\]](#)
59. Sarikhani, R.; Keynia, F. Cooperative Spectrum Sensing Meets Machine Learning: Deep Reinforcement Learning Approach. *IEEE Commun. Lett.* **2020**, *24*, 1459–1462. [\[CrossRef\]](#)
60. Cai, P.; Zhang, Y.; Pan, C. Coordination Graph-Based Deep Reinforcement Learning for Cooperative Spectrum Sensing Under Correlated Fading. *IEEE Wirel. Commun. Lett.* **2020**, *9*, 1778–1781. [\[CrossRef\]](#)
61. Gao, A.; Du, C.; Ng, S.X.; Liang, W. A Cooperative Spectrum Sensing with Multi-Agent Reinforcement Learning Approach in Cognitive Radio Networks. *IEEE Commun. Lett.* **2021**, *25*, 2604–2608. [\[CrossRef\]](#)
62. Ngo, Q.T.; Jayawickrama, B.A.; He, Y.; Dutkiewicz, E. Multi-Agent DRL-Based RIS-Assisted Spectrum Sensing in Cognitive Satellite–Terrestrial Networks. *IEEE Wirel. Commun. Lett.* **2023**, *12*, 2213–2217. [\[CrossRef\]](#)
63. Ali, A.; Hamouda, W. Advances on Spectrum Sensing for Cognitive Radio Networks: Theory and Applications. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 1277–1304. [\[CrossRef\]](#)
64. Arjoune, Y.; Kaabouch, N. A Comprehensive Survey on Spectrum Sensing in Cognitive Radio Networks: Recent Advances, New Challenges, and Future Research Directions. *Sensors* **2019**, *19*, 126. [\[CrossRef\]](#)
65. Zhang, Y.; Luo, Z. A Review of Research on Spectrum Sensing Based on Deep Learning. *Electronics* **2023**, *12*, 4514. [\[CrossRef\]](#)
66. Wijenayake, C.; Madanayake, A.; Kota, J.; Bruton, L. Space-Time Spectral White Spaces in Cognitive Radio: Theory, Algorithms, and Circuits. *IEEE J. Emerg. Sel. Top. Circuits Syst.* **2013**, *3*, 640–653. [\[CrossRef\]](#)
67. Madanayake, A.; Wijenayake, C.; Tran, N.; Cooklev, T.; Hum, S.; Bruton, L.T. Directional spectrum sensing using tunable multi-D space-time discrete filters. In Proceedings of the IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, San Francisco, CA, USA, 25–28 June 2012; pp. 1–6.

68. Wijenayake, C.; Madanayake, A.; Bruton, L.T.; Devabhaktuni, V. DOA-estimation and source-localization in CR-networks using steerable 2-D IIR beam filters. In Proceedings of the IEEE International Symposium on Circuits and Systems, Beijing, China, 19–23 May 2013; pp. 65–68.
69. Gunaratne, T.K.; Bruton, L.T. Adaptive complex-coefficient 2D FIR trapezoidal filters for broadband beamforming in cognitive radio systems. *Circuits Syst. Signal Process.* **2011**, *30*, 587–608. [\[CrossRef\]](#)
70. Wilcox, D.; Tsakalaki, E.; Kortun, A.; Ratnarajah, T.; Papadias, C.B.; Sellathurai, M. On Spatial Domain Cognitive Radio Using Single-Radio Parasitic Antenna Arrays. *IEEE J. Sel. Areas Commun.* **2013**, *31*, 571–580. [\[CrossRef\]](#)
71. Qian, R.; Sellathurai, M.; Ratnarajah, T. Directional spectrum sensing for cognitive radio using ESPAR arrays with a single RF chain. In Proceedings of the 2014 European Conference on Networks and Communications (EuCNC), Bologna, Italy, 23–26 June 2014; pp. 1–5.
72. Yazdani, H.; Vosoughi, A. On cognitive radio systems with directional antennas and imperfect spectrum sensing. In Proceedings of the 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), New Orleans, LA, USA, 5–9 March 2017; pp. 3589–3593.
73. Pulipati, S.; Ariyaratna, V.; Edussooriya, C.U.S.; Wijenayake, C.; Wang, X.; Madanayake, A. Real-Time FPGA-Based Multi-Beam Directional Sensing of 2.4 GHz ISM RF Sources. In Proceedings of the 2019 Moratuwa Engineering Research Conference (MERCon), Moratuwa, Sri Lanka, 3–5 July 2019; pp. 129–134.
74. Richard Thompson, A.; Gergely, T.E.; Vanden Bout, P.A. Interference and Radioastronomy. *Phys. Today* **1991**, *44*, 41–49. [\[CrossRef\]](#)
75. Science Goals-Square Kilometre Array Observatory. Available online: <https://www.skao.int/en/explore/science-goals> (accessed on 27 May 2024).
76. Carpenter, J.; Brogan, C.; Iono, D.; Mroczkowski, T. The ALMA2030 Wideband Sensitivity Upgrade. *arXiv* **2022**, arXiv:2211.00195.
77. Science with a Next-Generation very Large Array. Available online: <https://ngvla.nrao.edu/page/science> (accessed on 27 May 2024).
78. Sihlangu, I.; Oozeer, N.; Bassett, B.A. Nature and Evolution of UHF and L-band Radio Frequency Interference at the MeerKAT Radio Telescope. In Proceedings of the RFI Workshop 2022, European Centre for Medium-Range Weather Forecasts (ECMWF), Online, 14–18 February 2022.
79. Galt, J. Contamination from satellites. *Nature* **1990**, *345*, 483–483. [\[CrossRef\]](#)
80. De Pree, C.G.; Anderson, C.R.; Zheleva, M. 2023. Astronomy Is under Threat by Radio Interference from Satellites—Here’s What Can Be Done about It. Available online: <https://www.weforum.org/agenda/2023/03/astronomy-radio-interference-satellites-technology/> (accessed on 27 May 2024).
81. Preferred frequency bands for radio astronomical measurements below 1 THz. Recommendation ITU-R RA.314-11, International Telecommunication Union—Radiocommunication Sector. 2023. Available online: https://www.itu.int/dms_pubrec/itu-r/rec/ra/R-REC-RA.314-11-202312-I!!PDF-E.pdf (accessed on 27 May 2024).
82. Protection Criteria Used for Radio Astronomical Measurements. Recommendation ITU-R RA.769-2, International Telecommunication Union—Radiocommunication Sector. 2003. Available online: https://www.itu.int/dms_pubrec/itu-r/rec/ra/R-REC-RA.769-2-200305-I!!PDF-E.pdf (accessed on 27 May 2024).
83. Fridman, P.A.; Baan, W.A. RFI mitigation methods in radio astronomy. *Astron. Astrophys.* **2001**, *378*, 327–344. [\[CrossRef\]](#)
84. Zhang, L.; Jin, R.; Zhang, Q.; Wang, R.; Zhang, H.; Wen, Z. Fusion Method of RFI Detection, Localization, and Suppression by Combining One-Dimensional and Two-Dimensional Synthetic Aperture Radiometers. *Remote Sens.* **2024**, *16*, 667. [\[CrossRef\]](#)
85. Proakis, J.G.; Manolakis, D.G. *Digital Signal Processing*; Pearson Prentice Hall: Upper Saddle River, NJ, USA, 2007.
86. Oppenheim, A.V.; Schaffer, R.W.; Buck, J. *Discrete-Time Signal Processing*; Prentice Hall International Editions; Prentice Hall: Upper Saddle River, NJ, USA, 1999.
87. Baghaie, R.; Dimitrov, V. DHT algorithm based on encoding algebraic integers. *Electron. Lett.* **1999**, *35*, 1303–1305. [\[CrossRef\]](#)
88. Britanak, V.; Yip, P.; Rao, K.R. *Discrete Cosine and Sine Transforms*; Academic Press: Cambridge, MA, USA, 2007.
89. Feig, E.; Winograd, S. On the multiplicative complexity of discrete cosine transforms. *IEEE Trans. Inf. Theory* **1992**, *38*, 1387–1391. [\[CrossRef\]](#)
90. Malvar, H.S.; Hallapuro, A.; Karczewicz, M.; Kerofsky, L. Low-complexity transform and quantization in H.264/AVC. *IEEE Trans. Circuits Syst. Video Technol.* **2003**, *13*, 598–603. [\[CrossRef\]](#)
91. Winograd, S. *Arithmetic Complexity of Computations*; CBMS-NSF Regional Conference Series in Applied Mathematics; SIAM: Philadelphia, PA, USA, 1980; Volume 33.
92. Winograd, S. On Computing the Discrete Fourier Transform. *Math. Comput.* **1978**, *32*, 175–199. [\[CrossRef\]](#)
93. Heideman, M.T. *Multiplicative Complexity, Convolution, and the DFT*; Springer: Heidelberg, Germany, 1988.
94. Oppenheim, A.V.; Verghese, G.C. *Signals, Systems & Inference*; Prentice-Hall Signal Processing Series; Pearson: Upper Saddle River, NJ, USA, 2016.
95. Cintra, R.J. An integer approximation method for discrete sinusoidal transforms. *Circuits Syst. Signal Process.* **2011**, *30*, 1481–1501. [\[CrossRef\]](#)
96. Suárez Villagrán, D.M. Discrete Fourier Transform Approximations with Applications in Detection and Estimation. Master’s Thesis, Universidade Federal de Pernambuco, Recife, Brazil, 2015.

97. Madanayake, A.; Ariyaratna, V.; Madishetty, S.; Pulipati, S.; Cintra, R.J.; Coelho, D.; Oliveira, R.; Bayer, F.M.; Belostotski, L.; Mandal, S.; et al. Towards a Low-SWaP 1024-Beam Digital Array: A 32-Beam Subsystem at 5.8 GHz. *IEEE Trans. Antennas Propag.* **2020**, *68*, 900–912. [\[CrossRef\]](#)
98. Madanayake, A.; Cintra, R.J.; Akram, N.; Ariyaratna, V.; Mandal, S.; Coutinho, V.A.; Bayer, F.M.; Coelho, D.; Rappaport, T.S. Fast Radix-32 Approximate DFTs for 1024-Beam Digital RF Beamforming. *IEEE Access* **2020**, *8*, 96613–96627. [\[CrossRef\]](#)
99. Blahut, R.E. *Fast Algorithms for Signal Processing*; Cambridge University Press: Cambridge, UK, 2010. [\[CrossRef\]](#)
100. The Collaboration for Astronomy Signal Processing and Electronics Research. 2024. <https://casper.berkeley.edu/> (accessed on 23 May 2024).
101. The CASPER Toolflow. GitHub. 2024. https://github.com/casper-astro/mlib_devel/ (accessed on 24 May 2024).
102. Stutzman, W.L.; Thiele, G.A. *Antenna Theory and Design*; Wiley: Hoboken, NJ, USA, 2012.
103. Kulasekera, S.; Madanayake, A.; Suarez, D.; Cintra, R.J.; Bayer, F.M. Multi-beam receiver apertures using multiplierless 8-point approximate DFT. In Proceedings of the 2015 IEEE Radar Conference (RadarCon), Arlington, VA, USA, 10–15 May 2015; pp. 1244–1249. [\[CrossRef\]](#)
104. Zhao, H.; Madanayake, A.; Cintra, R.J.; Mandal, S. Analog Current-Mode 8-Point Approximate-DFT Multi-Beamformer with 4.7 Gbps Channel Capacity. *IEEE Access* **2023**, *11*, 53716–53735. [\[CrossRef\]](#)
105. Kulasekera, S.; Madanayake, A.; Wijenayake, C.; Bayer, F.M.; Suarez, D.; Cintra, R.J. Multi-beam 8×8 RF aperture digital beamformers using multiplierless 2-D FFT approximations. In Proceedings of the 2015 Moratuwa Engineering Research Conference (MERCon), Moratuwa, Sri Lanka, 7–8 April 2015; pp. 260–264. [\[CrossRef\]](#)
106. Ariyaratna, V.; Madanayake, A.; Tang, X.; Coelho, D.; Cintra, R.J.; Belostotski, L.; Mandal, S.; Rappaport, T.S. Analog Approximate-FFT 8/16-Beam Algorithms, Architectures and CMOS Circuits for 5G Beamforming MIMO Transceivers. *IEEE J. Emerg. Sel. Top. Circuits Syst.* **2018**, *8*, 466–479. [\[CrossRef\]](#)

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.

Complete Subhedge Projection for Stepwise Hedge Automata [†]

Antonio Al Serhali * and Joachim Niehren * 

Inria Center, University of Lille, 59000 Lille, France

* Correspondence: antonio.al-serhali@inria.fr (A.A.S.); joachim.niehren@inria.fr (J.N.)

[†] This paper is an extended version of our paper published in International Symposium on Fundamentals of Computation Theory, Trier, Germany, 18–21 September 2023.

Abstract: We demonstrate how to evaluate stepwise hedge automata (SHAs) with subhedge projection while completely projecting irrelevant subhedges. Since this requires passing finite state information top-down, we introduce the notion of downward stepwise hedge automata. We use them to define in-memory and streaming evaluators with complete subhedge projection for SHAs. We then tune the evaluators so that they can decide on membership at the earliest time point. We apply our algorithms to the problem of answering regular XPath queries on XML streams. Our experiments show that complete subhedge projection of SHAs can indeed speed up earliest query answering on XML streams so that it becomes competitive with the best existing streaming tools for XPath queries.

Keywords: automata; projection algorithm; streaming algorithm; XML

1. Introduction

Hedges are sequences of letters from the alphabet and trees $\langle h \rangle$ where h is again a hedge. Hedges are abstract from the syntactical details of XML or JSON documents while still being able to represent them. The linearization of a hedge is a nested word that can be written in a file. In this article, we study the problem of hedge pattern matching, i.e., whether a given hedge h belongs to a given regular hedge language L . Regular hedge languages can be defined in XPath or JSONPath in particular.

Projection is necessary for the efficiency of many algorithms on words, trees, hedges, or nested words. Intuitively, an algorithm is projecting if it visits only a fragment of the input structure: in the best case, the part that is relevant to the problem under consideration. The relevance of projection for XML processing was already noticed by [1–4]. Saxon’s in-memory evaluator, for instance, projects input XML documents relative to an XSLT program, which contains a collection of XPath queries to be answered simultaneously [5]. The QuiXPath tool [4] evaluates XPath queries in streaming mode with subtree and descendant projection. Projection during the evaluation of JSONPath queries on JSON documents in streaming mode is called fast-forwarding [6].

Projecting in-memory evaluation assumes that the full graph of the input hedge is constructed beforehand. Nevertheless, projection may still save time if one has to match several patterns on the same input hedge or if the graph was constructed for different reasons anyway. In streaming mode, the situation is similar: the whole input hedge on the stream needs to be parsed, but the evaluators need to inspect only nodes that are not projected away. Given that pure parsing is by two or three orders of magnitude faster than pattern evaluation, the time gain of projection may be considerable.

The starting point of this article is the notion of the subhedge irrelevance of positions of hedge h with respect to a hedge language L that we introduce. These are positions of h where, for all possible continuations, inserting any subhedge does not affect membership to L . We contribute an algorithm for hedge pattern matching with complete subhedge projection. Our algorithm decides hedge pattern matching while completely projecting away the subhedges located at irrelevant subhedge positions. In other words, it decides



Citation: Al Serhali, A.; Niehren, J. Complete Subhedge Projection for Stepwise Hedge Automata. *Algorithms* **2024**, *17*, 339. <https://doi.org/10.3390/a17080339>

Academic Editor: Henning Fernau and Klaus Jansen

Received: 29 January 2024

Revised: 23 July 2024

Accepted: 23 July 2024

Published: 2 August 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

whether a hedge h matches a pattern L without visiting any subhedge of h that is located at a position that is irrelevant with respect to L .

Regular hedge languages can be represented by nested regular expressions, which can be derived from regular XPath queries [7] or by some kind of hedge automata, which can be compiled from nested regular expressions. We use stepwise hedge automata (SHAs) [8], a recent variant of standard hedge automata, which, in turn, date back to the 1960s [9,10]. SHAs mix up the bottom-up processing of standard tree automata with the left-to-right processing of finite word automata (DFAs). They neither support top-down processing nor have an explicit stack, unlike nested word automata NWAs [11–15]. SHAs have a good notion of bottom-up and left-to-right determinism, avoiding the problematic notion of determinism for standard hedge automata and the too-costly notion of determinism for NWAs. Furthermore, compilers from nested regular expressions to SHAs are available [8], and determinization algorithms [16] produce small SHAs for all regular XPath queries in the existing practical benchmarks [7,17].

The motivation for the present work is to add subhedge projection to a recent streaming algorithm for deterministic SHAs that can evaluate regular XPath queries in the earliest manner. Most alternative streaming approaches avoid earliest query answering by considering sublanguages of regular queries without delays so that node selection depends only on the past of the node but not on the future [18,19] or by admitting late selection [20]. In particular, it was recently demonstrated that the earliest query answering for regular monadic queries defined by deterministic SHAs [21] has a lower worst-case complexity than for deterministic NWAs [22]. Thereby, the earliest query answering for regular queries became feasible in practice for the first time. On the other hand side, it is still slower experimentally than the best existing non-earliest approaches for general regular monadic queries on XML streams (see [23] for a comparison) since the latter supports projection. How projections could be made for SHAs has not been studied before the conference version of the present article [24].

Consider the example of the XPath filter `[self::list/child::item]` that is satisfied by an XML document if its root is a `list` element that has an `item` child. With the encoding of the XML document as hedges are chosen in the present article, satisfying hedges have the form $\langle \text{list} \cdot h_1 \cdot \langle \text{item} \cdot h_2 \rangle \cdot h_3 \rangle \cdot h_4$, and for some hedges, h_1, h_2, h_3, h_4 . When evaluating this filter on a hedge, it is sufficient to inspect the first subtree for having the root label `list` and then all its children until one with a root label `item` is found. The subhedges of these children, i.e., the hedge h_3 and the subhedges of the top-level trees in h_1 and h_3 , can be projected away, as well as the hedge h_4 : they do not have to be inspected for evaluating this filter. However, one must memorize whether the level of the current node is 0, 1, or greater. This level of information can be naturally updated in a top-down manner. The evaluators of SHAs, however, operate bottom-up and left-to-right exclusively. Therefore, projecting evaluators for SHAs need to be based on more general machines. We propose *downward stepwise hedge automata* ($\text{SHA}^\downarrow$ s), a variant of SHAs that supports top-down processing. They are basically Neumann and Seidl's pushdown forest automata [25], except that they are applied to unlabeled hedges instead of labeled forests. Furthermore, NWAs are known to operate in basically the same manner on nested words [26] while allowing for slightly more general visible pushdowns. We then distinguish subhedge projection states for $\text{SHA}^\downarrow$ s, and show how to use them to evaluate SHAs with subhedge projection both in memory and in streaming mode.

As a first contribution, we present the safe-no-change compiler from SHAs to $\text{SHA}^\downarrow$ s that can distinguish appropriate subhedge projection states. The idea is that the safe-no-change $\text{SHA}^\downarrow$ can distinguish contexts in which the states of the SHA will safely not change. For instance, the XPath filter `[self::list/child::item]` can be defined by a deterministic SHA operating bottom-up which our compiler maps to the $\text{SHA}^\downarrow$ operating top-down. The context information made explicit by top-down processing is about the levels of the states. This permits us to distinguish projection states starting from level 2, and in which subhedges can be ignored. We prove the soundness of our compiler based on a nontrivial invariant that

we establish (we note that the proof required an adaptation of the original compiler from FCT [24]). We also present a counter-example showing that the safe-no-change projection is not complete for subhedge projection. It shows that a subhedge may be irrelevant even though its state may still be changing.

As a second contribution, we propose the congruence projection algorithm. It again compiles SHAs to $\text{SHA}^\downarrow$ s but relies on the congruence relations of automata states. We then prove that congruence projection yields not only a sound algorithm but that this algorithm is also complete for subhedge projection, i.e., all strongly irrelevant subhedges (see Definition 9) are evaluated into a looping state. Congruence projection starts on the top-level of hedges with the Myhill–Nerode congruence that is well-known from automata minimization.

For languages with words L , this congruence identifies prefixes v that have the same residual language $v^{-1}(L) = \{w \mid v \cdot w \in L\}$. A prefix v is then an irrelevant suffix if its residual language $v^{-1}(L)$ is either universal or empty. In the case of hedges, which extend on words by adding hierarchical nesting via a tree constructor, the congruence must be adapted when moving down into subtrees.

We show that both compilers may increase the size of the automata exponentially since they must be able to convert bottom-up deterministic automata on monadic trees into top-down deterministic automata. That is the same as inverting deterministic automata on words, which is well known to raise an exponential blow-up in the worst case (e.g., for the family of languages $(a + b)^* \cdot a \cdot (a + b)^n$, where $n \in \mathbb{N}$, the minimal deterministic left-to-left automaton, has 2^{n+1} states while the minimal deterministic right-to-left automaton has $n + 1$ states).

The exponential explosion can be avoided when interested in the evaluation of hedges by automaton with subhedge projection, by not constructing the complete $\text{SHA}^\downarrow$ s statically. Instead, only the required part of the $\text{SHA}^\downarrow$ s may be constructed on the fly when evaluating a hedge.

Our third contribution is the refinement of the two previous compilers so that they can also distinguish safe states for selection at the earliest position. For this, we combine these compilers with a variant of the earliest membership tester of [21] that operates in memory by compiling $\text{SHA}^\downarrow$ s instead of NWAs. Furthermore, membership failure is detected at the earliest position, too. In this way, we obtain the earliest in-memory membership tester for deterministic SHAs.

Our fourth contribution is to show how to run the previous in-memory evaluators in streaming mode while reading the linearization of the hedge into a nested word as an input. Thereby, we can improve on the recent earliest streaming membership tester behind the AStream tool [21] by adding subhedge projection to it.

Our fifth and last contribution is an implementation and experimental evaluation of the streaming earliest query answering algorithm for dSHAs by introducing subhedge projection into the AStream tool [21]. We implemented both safe-no-change and congruence projection. For this, we lifted our earliest membership tester with subhedge projection to an earliest query answering algorithm for monadic queries. For the experimentation, we started with deterministic SHAs constructed with the compiler from [8] for the forward regular XPath queries of the XPathMark benchmark [27] and real-world XPath queries [7]. It turns out that congruence projection projects much more strongly than the safe-no-change projection for at least half of the benchmark queries. It reduces the running time for all regular XPath queries considerably since large parts of the input hedges can be projected away. In our benchmark, the projected percentage ranges from 75.7% to 100% of the input stream. For XPath queries that contain only child axes, the earliest query answering algorithm of AStream with congruence projection becomes competitive in efficiency with the best existing streaming tool called QuiXPath [23], which is not always earliest for some queries. Our current implementation of the earliest congruence projection in AStream is by a factor from 1.3 to 2.9 slower than QuiXPath on the benchmark queries. The improvement is smaller for XPath queries with the descendant axis, where less subhedge projection is

possible. Instead, some kind of descendant projection would be needed. Still, even in the worst case in our benchmark, the earliest congruence projection with AStream is currently only by a factor of 13.8 slower than QuiXPath.

Outline. We start with preliminaries on hedges and nested words in Section 2. In Section 3, we recall the problem of hedge pattern matching and formally define irrelevant subhedges. In Section 4, we recall the definition of SHAs, introduce $\text{SHA}^\downarrow$ s, and discuss their in-memory evaluators. In Section 5, we introduce the notion of subhedge projection states, define when an $\text{SHA}^\downarrow$ is complete for subhedge projection, and present an in-memory evaluator for $\text{SHA}^\downarrow$ s with subhedge projection. In Section 6, we introduce the safe-no-change projection, state its soundness, and show its incompleteness. In Section 7, we introduce congruence projection and prove it to be sound and complete for subhedge projection. The earliest in-memory evaluators for $\text{SHA}^\downarrow$ s with subhedge projection follow in Section 8. Streaming variants of our in-memory evaluators are derived systematically in Section 9. Section 10 discusses how to lift our algorithms from membership testing to monadic queries. Section 11 discusses our practical experiments. Section 12 discusses further related work on automata notions related to SHAs and $\text{SHA}^\downarrow$ s. Appendix A presents the reformulation of the earliest query-answering algorithm from [21] based on $\text{SHA}^\downarrow$ s and with schema restrictions. Appendix B provides sound proof for the safe-no-change projection.

Publication Comments. This original version of the journal article was published at the FCT conference [24]. Compared to the previous publication, the following contributions are new:

- The definition of subhedge irrelevant prefixes of nested words and the definition of completeness for subhedge projection (Section 3).
- The addition of schema restrictions throughout the paper and the notion of schema completeness (Section 4.3).
- The notion of completeness for subhedge projection (Section 5.2).
- The soundness proof of the safe-no-change projection algorithm in Section 6.2, moved to Appendix B.
- The congruence projection algorithm and the proof of its soundness and completeness for subhedge projection (Section 7).
- A systematic method to add subhedge projection to an earliest $\text{SHA}^\downarrow$ (Section 8).
- A systematic method to reduce the correctness of streaming automata evaluators to the corresponding in-memory evaluators (Section 9).
- A discussion of how to deal with monadic queries while exploiting the availability of schema restrictions (Section 10).
- An implementation of congruence projection with experimental results for XPath evaluation on XML streams (Section 11).
- A longer discussion on related work (Section 12).
- In Appendix A, we also show how to obtain the earliest $\text{dSHA}^\downarrow$ s from dSHAs by adapting the compiler from [21] mapping dSHAs to the earliest dNWAs.

2. Preliminaries

Let A and B be sets. If A is a subset of B , then the complement of A in B is denoted by $\overline{A} = B \setminus A$ while keeping the set relative to which the complement is taken implicit. The *domain* of the binary relation $r \subseteq A \times B$ is $\text{dom}(r) = \{a \in A \mid \exists b \in B. (a, b) \in r\}$. A *partial function* $f : A \hookrightarrow B$ is a binary relation $f \subseteq A \times B$ that is functional. A *total function* $f : A \rightarrow B$ is a partial function $f : A \hookrightarrow B$ with $\text{dom}(f) = A$.

Words. Let $\mathbb{N}$ be the set of natural numbers, including 0. Let the alphabet Σ be a set. The set of words over Σ is $\Sigma^* = \bigcup_{n \in \mathbb{N}} \Sigma^n$. A word is $(a_1, \dots, a_n) \in \Sigma^n$, where $n \in \mathbb{N}$ is written as $a_1 \dots a_n$. We denote the empty word of length 0 by $\varepsilon \in \Sigma^0$, and by $v_1 \cdot v_2 \in \Sigma^*$ the concatenation of two words $v_1, v_2 \in \Sigma^*$.

If $v = u \cdot v' \cdot w$ is a word, then we call u a prefix of v and v' a factor of v and w a suffix of v . Given any subset $L \subseteq \Sigma^*$, we denote the set of prefixes of words in L by $\text{prefs}(L)$ and the set of suffixes of words in L by $\text{suffs}(L)$.

For any subalphabet $\Sigma' \subseteq \Sigma$, the projection of a word $v \in \Sigma^*$ to Σ' is the word $\text{proj}_{\Sigma'}(v)$ in $(\Sigma')^*$ that is obtained from v by removing all letters from $\Sigma \setminus \Sigma'$.

Hedges. Hedges are sequences of letters and trees $\langle h \rangle$ with a hedge h . More formally, a hedge $h \in \mathcal{H}_\Sigma$ has the following abstract syntax:

$$h, h' \in \mathcal{H}_\Sigma ::= \varepsilon \mid a \mid \langle h \rangle \mid h \cdot h' \quad \text{where } a \in \Sigma$$

We assume $\varepsilon \cdot h = h \cdot \varepsilon = h$ and $(h \cdot h_1) \cdot h_2 = h \cdot (h_1 \cdot h_2)$. Therefore, we consider any word in Σ^* as a hedge in $\mathcal{H}_\Sigma$, i.e., $\Sigma^* \ni aab = a \cdot a \cdot b \in \mathcal{H}_\Sigma$. Any hedge can be converted or stored as a graph. For the signature $\Sigma = \{\text{list}, \text{item}, A, \dots, Z\}$, the graph of an example hedge in $\mathcal{H}_\Sigma$ is shown in Figure 1. It encodes the sequence of XML documents in Figure 2:

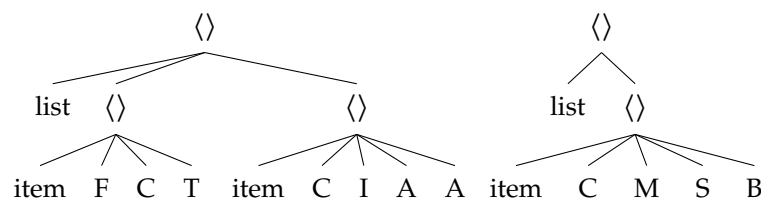


Figure 1. The graph of the hedge $\langle \text{list} \cdot \langle \text{item} \cdot F \cdot C \cdot T \rangle \cdot \langle \text{item} \cdot C \cdot I \cdot A \cdot A \rangle \rangle \cdot \langle \text{list} \cdot \langle \text{item} \cdot C \cdot M \cdot S \cdot B \rangle \rangle$ is a sequence of trees.

```
<list>
  <item>FCT</item>
  <item>CIAA</item>
</list>
<list>
  <item>CMSB</item>
</list>
```

Figure 2. The sequence of XML documents encoded by the hedge in Figure 1.

Nested Words. A nested word over Σ is a word over the alphabet $\hat{\Sigma} = \Sigma \cup \{\langle, \rangle\}$ in which all parentheses are well-nested. Intuitively, this means that for any opening parenthesis, there is a matching closing parenthesis and vice versa.

The set of nested words over Σ can be defined as the subsets of words over $\hat{\Sigma}$ that are a linearization of the hedge, where the linearization function $nw : \mathcal{H}_\Sigma \rightarrow (\Sigma \cup \{\langle, \rangle\})^*$ is defined as follows:

$$nw(\varepsilon) = \varepsilon, \quad nw(\langle h \rangle) = \langle \cdot nw(h) \cdot \rangle, \quad nw(a) = a, \quad nw(h \cdot h') = nw(h) \cdot nw(h').$$

So, the set of all nested words over Σ is $nw(\mathcal{H}_\Sigma)$. Let hdg be the inverse of the injective function nw restricted to its image, so the mapping from nested words to hedges with $hdg(nw(h)) = h$ for all $h \in \mathcal{H}_\Sigma$.

Nested Word Prefixes. Prefixes, suffixes, and factors of nested words may not be nested words themselves. For instance, the hedge $h = a \cdot \langle b \cdot c \rangle$ has the linearization $nw(h) = a \langle bc \rangle$. Its prefix $a \langle b$ is not well-nested since it has a dangling opening parenthesis. Its suffix $c \rangle$ is also not well-nested since it has a dangling closing parenthesis.

Any algorithm that traverses a hedge h in a top-down and left-to-right manner inspects all the prefixes of $nw(h)$. Any prefix v of $nw(h)$ not only distinguishes the position of the hedge h but also specifies the part of the hedge that is located before this position in the linearization $nw(h)$. An example is illustrated graphically in Figure 3. This particularly holds for streaming algorithms that receive the nested word $nw(h)$ as an input on a stream

and may be inspected only once from the left to the right. But it holds equally for in-memory algorithms that receive the input hedge h as a hierarchical structure whose graph is stored in memory and then traverses the graph top-down and left-to-right. Any $\langle \rangle$ node will then be visited twice: once when reading an opening parenthesis $\langle$ and going down into a subhedge, and another time when going up to the closing parenthesis $\rangle$ after having processed the subhedge.

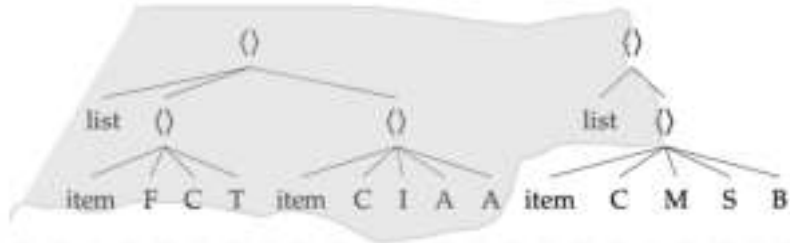


Figure 3. The part of the hedge distinguished by the nested word prefix $\langle list \cdot \langle item \cdot F \cdot C \cdot T \rangle \cdot \langle item \cdot C \cdot I \cdot A \cdot A \rangle \rangle \cdot \langle list \cdot \langle$.

3. Problem

We formally define the pattern-matching problem for hedges and the notion of sub-hedge irrelevance, which allows us to characterize the parts of the input hedge that do not need to be visited during pattern matching.

3.1. Hedge Pattern Matching

We are interested in algorithms that match a hedge pattern against a hedge. We start with algorithms that test whether such matching exists. This is sometimes called filtering or, alternatively, Boolean query answering. In Section 10, we consider the more general problem of returning the set of matchings. This problem is sometimes called monadic query answering. Our experiments in Section 11 apply monadic query answering for regular XPath queries.

We start from hedge patterns that are nested regular expressions with letters in the signature Σ . For these, we fix a set V of recursion variables and assume that it is disjoint from Σ .

$$e, e' \in nRegExp_{\Sigma} ::= \varepsilon \mid a \mid z \mid e \cdot e' \mid e + e' \mid e^* \mid \langle e \rangle \mid \mu z.e \quad \text{where } a \in \Sigma, z \in V$$

Each nested regular expression $e \in nRegExp_\Sigma$ is satisfied by a subset of hedges in $\llbracket e \rrbracket \in \mathcal{H}_{\Sigma \cup V}$. We recall the definition of this set only informally. The expression ε is satisfied only by the empty word, expression $a \in \Sigma$ only by the hedge a only, and expression $z \in V$ only by the one letter hedge z . An expression $e \cdot e'$ with the concatenation operator and $e, e' \in nRegExp_\Sigma$ is satisfied by the concatenations of hedges satisfying e and e' , respectively. An expression e^* with Kleene's star operator and $e \in nRegExp_\Sigma$ is satisfied by repetitions of words satisfying e . The Kleene star provides for horizontal recursion on hedges. The μ expression $\mu z.e$, where $z \in V$ and $e \in nRegExp_\Sigma$, is satisfied by all hedges without recursion variable x in which z is repeatedly replaced by a hedge satisfying e until it disappears. We adopt the usual restriction [28] that in any nested regular expression $\mu z.e$, all occurrences of z in e are guarded by a tree constructor $\langle . \rangle$. In this way, the μ -operator provides proper vertical recursion. Also note that, for defining sublanguages of $\mathcal{H}_\Sigma$, we only need such expressions in $nRegExp_\Sigma$ in which all occurrences of recursion variables z are bound in the scope of binder μz .

It is well known that nested regular expressions in $nRegExp_\Sigma$ can define all regular sublanguages of hedges in $\mathcal{H}_\Sigma$, i.e., they have the same expressiveness as hedge automata with alphabet Σ . Analogous results for regular tree expressions and tree automata are folklore [29].

Example 1 (Nested regular expressions of regular XPath filters). For instance, we can define subsets of hedge encodings of XML documents with two kinds of elements in $\Sigma = \{\text{list}, \text{item}\}$ by expressions in $n\text{RegExp}_\Sigma$. For this article, we will use an encoding of the XML document, which produces hedges that satisfy the following regular expression $\text{doc} \in n\text{RegExp}_\Sigma$, where $z \in V$:

$$\text{doc} =_{\text{def}} \text{tree}^* \quad \text{where} \quad \text{tree} =_{\text{def}} \mu z. \langle (\text{list} + \text{item}) \cdot z^* \rangle$$

In our application, the encodings are a little richer in order to distinguish different types of XML nodes such as elements, attributes, comments, and texts. An XPath filter such as $[\text{self}::\text{list}/\text{child}::\text{item}]$ can then be expressed by the nested regular expression:

$$\text{filter} =_{\text{def}} \langle \text{list} \cdot \text{doc} \cdot \langle \text{item} \cdot \text{doc} \rangle \cdot \text{doc} \rangle \cdot \text{doc}$$

General compilers from regular downward XPath filter queries to nested regular expressions were proposed in [8]. An implementation was developed in the context of the AStream tool [21].

We will use schemas $S \subseteq \mathcal{H}_\Sigma$ to restrict the inputs of our pattern-matching problem. As we are mostly interested in regular schemas, we do not assume regularity globally since safe-no-change projection does not rely on it. The pattern-matching problem for nested regular expressions with respect to a schema $S \subseteq \mathcal{H}_\Sigma$ receives the following inputs:

- a nested regular expression $e \in n\text{RegExp}_\Sigma$ and
- a hedge $h \in S$.

It then returns the truth value of the judgment $h \in \llbracket e \rrbracket$. The input hedge $h \in S$ may either be given by a graph that resides in memory or by a stream containing the linearization $nw(h)$ of the input hedge, which can be read only once from left to right.

3.2. Irrelevant Subhedges

We define the concept of irrelevant occurrences of subhedges with respect to a given hedge pattern. What this means depends on the kind of algorithm that we will use for pattern matching. We will use algorithms that operate on the input hedge top-down, left-to-right, and bottom-up. This holds for streaming algorithms in particular.

Intuitively, when the pattern-matching algorithm reaches a node top-down or left-to-right whose subsequent subhedge is irrelevant, then it can jump over it without inspecting its structure. What jumping means should be clear if the hedge is given by a graph that is stored in memory. Notice that the full graph is to be constructed beforehand, even though some parts of it may turn out irrelevant. Still, one may win lots of time by jumping over irrelevant parts if either the graph was already constructed for other reasons or if many pattern-matching problems are to be solved on the same hedge.

In the case of streams, the irrelevant subhedge still needs to be parsed, but it will not be analyzed otherwise. Most typically, the possible analysis is carried out by automata, which may take two orders of magnitude more time than needed for parsing. Therefore, not having to do any analysis may considerably speed up the streaming algorithm.

Definition 1. Let $S \subseteq \mathcal{H}_\Sigma$ be a schema and $L \subseteq S$ a language satisfying this schema. We define diff_S^L as the least symmetric relation on prefixes of nested words $u, u' \in \text{pref}_S(\mathcal{N}_\Sigma)$:

$$\text{diff}_S^L(u, u') \Leftrightarrow \exists w \in \text{suff}_S(\mathcal{N}_\Sigma). u \cdot w \in nw(L) \wedge u' \cdot w \in nw(S \setminus L).$$

A nested word prefix u is called subhedge relevant for L with schema S if there exists nested words $v, v' \in \mathcal{N}_\Sigma$ such that $\text{diff}_S^L(u \cdot v, u \cdot v')$. Otherwise, the prefix u is called subhedge irrelevant for L with schema S .

So, $\text{diff}_S^L(u, u')$ states that u and u' have continuations that behave differently with respect to L and S . Furthermore, a prefix u is subhedge irrelevant if language membership does not depend on hedge insertion at u under the condition that schema membership is

guaranteed. The case without schema restrictions is subsumed by the above definition by choosing $S = \mathcal{H}_\Sigma$. In this case, the complement $\overline{\text{diff}_{\mathcal{H}_\Sigma}^L}$ is the Myhill–Nerode congruence of L , which is well-known in formal language theory from the minimization of DFAs (see any standard textbook on formal language theory or Wikipedia (Myhill–Nerode theorem: https://en.wikipedia.org/wiki/Myhill-Nerode_theorem, accessed on 23 July 2024)).

This congruence serves as the basis for Gold-style learning of regular languages from positive and negative examples (see [30] or Wikipedia (Gold-style learning in the limit https://en.wikipedia.org/wiki/Language_identification_in_the_limit, accessed on 23 July 2024)).

Schema restrictions are needed for our application to regular XPath patterns.

Example 2 (Regular XPath filters). Consider the schema $S = \llbracket \text{doc} \rrbracket$ for hedges representing XML documents with signature $\Sigma = \{\text{list}, \text{item}\}$, and the regular hedge language $L = \llbracket \text{filter} \rrbracket$, where

$$\text{filter} = \langle \text{list} \cdot \text{tree}^* \cdot \langle \text{item} \cdot \text{tree}^* \rangle \cdot \text{tree}^* \rangle \cdot \text{tree}^*$$

is the nested regular expression from Example 1 for the XPath filter `[self::list/child::item]`. Recall that this nested regular expression can be applied to hedges representing XML documents only. The nested word prefix $u = \langle \text{list} \cdot \langle \text{item} \rangle \rangle$ is subhedge irrelevant for the language L with schema S . Note that its continuation $\langle \text{list} \cdot \langle \text{item} \rangle \rangle$ with the suffix $w = \rangle \rangle$ belongs to L . Hence, for any $h \in \mathcal{H}_\Sigma$, if the continuation $\langle \text{list} \cdot \langle \text{item} \cdot h \rangle \rangle$ belongs to $S = \llbracket \text{doc} \rrbracket$, then it also belongs to L . Nevertheless, for the hedge $h_1 = \langle \rangle$, the continuation $\langle \text{list} \cdot \langle \text{item} \cdot h_1 \rangle \rangle$ does not belong to L since it does not satisfy the schema $S = \llbracket \text{doc} \rrbracket$.

The prefix $\langle \text{item} \rangle$ is also irrelevant for the language L , even independently of the schema. This is because L does not contain any continuation of this prefix to some hedge. The prefix $\langle \text{list} \rangle$ is not irrelevant for the language L wrt S . This can be seen with the suffix $w = \rangle$, since $\langle \text{list} \rangle$ does not belong to L while the continuation $\langle \text{list} \cdot h \rangle$ with $h = \langle \text{item} \rangle$ does belong to L and both continuations satisfy the schema $S = \llbracket \text{doc} \rrbracket$.

However, schema restrictions may also have surprising consequences that make the problem more tedious.

Example 3 (Surprising consequences of schema restrictions). Consider the signature $\Sigma = \{a\}$, the pattern $L = \{\langle a \rangle\}$, and the schema $S = L \cup \{\langle \rangle \cdot a\}$. The prefix $\langle \rangle$ is indeed subhedge irrelevant to L and S . In order to see this, we consider all possible closing suffixes one by one. The smallest closing prefix is $\rangle$. Note that a hedge $\langle h \rangle \in L$ if and only if $h = a$. So, membership to L seems to depend on the subhedge h . However, also note that $\langle h \rangle \in S$ if and only if $h = a$. Therefore, when assuming that $\langle h \rangle$ belongs to the schema S , it must also belong to L . So, the pattern must match when assuming schema membership. The next closing suffix is $\rangle \cdot a$. Note that a hedge $\langle h \rangle \cdot a \in S$ if and only if $h = \epsilon$. However, $\langle h \rangle \cdot a \notin L$, so the pattern will not match for any subhedge h with this prefix when assuming the input hedge satisfies the schema. The situation is similar for larger suffixes, so in no case does language membership depend on the subhedge at prefix $\langle \rangle$ when assuming that the full hedge satisfies the schema S .

3.3. Basic Properties

We first show that subhedge irrelevant prefixes remain irrelevant when extended by the nested word of any hedge. This property should be expected for any reasonable notion of subhedge irrelevance.

Lemma 1. For any nested word prefix u , language $L, S \subseteq \mathcal{H}_\Sigma$, and nested word $v \in \mathcal{N}_\Sigma$, if the prefix u is subhedge irrelevant for L with schema S , then the prefix $u \cdot v$ is so too.

Proof. Let u be subhedge irrelevant for L with schema S and $v \in \mathcal{N}_\Sigma$ a nested word. We fix arbitrary nested words $v', v'' \in \mathcal{N}_\Sigma$ and a nested word suffix $w \in \text{suffs}(\mathcal{N}_\Sigma)$ such that

$u \cdot v \cdot v' \cdot w \in nw(\mathbf{S})$ and $u \cdot v \cdot v'' \cdot w \in nw(\mathbf{S})$. Since $v \cdot v'$ and $v \cdot v''$ are also nested words, the subhedge irrelevance of u yields the following:

$$u \cdot v \cdot v' \cdot w \in nw(L) \Leftrightarrow u \cdot v \cdot v'' \cdot w \in nw(L).$$

Hence, the nested word prefix $u \cdot v$ is subhedge irrelevant for L with schema $\mathbf{S}$. $\square$

Definition 2. We call a binary relation D on prefixes of nested words a difference relation on prefixes if for all prefixes of nested words $u, u' \in \text{prefs}(\mathcal{N}_\Sigma)$ and nested words $v \in \mathcal{N}_\Sigma$:

$$(u \cdot v, u' \cdot v) \in D \Rightarrow (u, u') \in D.$$

Lemma 2. diff_S^L is a difference relation.

Proof. We must show for all prefixes $u, u' \in \text{prefs}(\mathcal{N}_\Sigma)$ and nested words $v \in \mathcal{N}_\Sigma$ that $(u \cdot v, u' \cdot v) \in \text{diff}_S^L$ implies $(u, u') \in \text{diff}_S^L$. So, let u, u' be prefixes of nested words and v be a nested word such that $(u \cdot v, u' \cdot v) \in \text{diff}_S^L$. Then, there exists a nested suffix w such that

$$u \cdot v \cdot w \in nw(L) \wedge u' \cdot v \cdot w \in nw(\mathbf{S} \setminus L).$$

The suffix $\tilde{w} = v \cdot w$ then satisfies $u \cdot \tilde{w} \in nw(L) \wedge u' \cdot \tilde{w} \in nw(\mathbf{S} \setminus L)$. Hence, $(u, u') \in \text{diff}_S^L$ as required. $\square$

In the schema-free case of words, the complement $\overline{\text{diff}_{\Sigma^*}^L}$ is an equivalence relation that is known as the Myhill–Nerode congruence. In the case of schemas, however, the complement $\overline{\text{diff}_S^L}$ may not be transitive; thus, it may not even be an equivalence relation. This might be surprising at first sight.

Example 4. Let $\Sigma = \{a, b\}$, $L = \{\varepsilon, b, aa\}$ and $\mathbf{S} = L \cup \{aab\}$. Then, $(\varepsilon, a) \in \overline{\text{diff}_S^L}$ since there is no continuation that extends both ε and a into the schema. For the same reason, we have $(a, aa) \in \overline{\text{diff}_S^L}$. However, $\text{diff}_S^L(\varepsilon, aa)$ since $\varepsilon \cdot b \in L$ and $aa \cdot b \in \mathbf{S} \setminus L$. Thus, $(\varepsilon, aa) \notin \overline{\text{diff}_S^L}$, so the complement $\overline{\text{diff}_S^L}$ is not transitive.

It should be noted for all languages L that $\overline{\text{diff}_{\mathcal{H}_\Sigma}^L}$ is reflexive and transitive, so it is an equivalence relation and thus a congruence. For general schemas $\mathbf{S}$, $\overline{\text{diff}_S^L}$ may not be transitive, as shown by Example 4, and may thus not be congruent.

This indicates that schemas may make it more difficult to decide subhedge irrelevance. Indeed, the natural manner in which one might expect to eliminate schemas based on complements does not work, as confirmed by following lemma (in particular, the counter-example in proof 1. $\nRightarrow$ 2.):

Lemma 3. For any nested word prefix $u \in \text{suffs}(\mathcal{N}_\Sigma)$ and $L, \mathbf{S} \subseteq \mathcal{H}_\Sigma$, consider the following two properties:

1. u is irrelevant for L with schema $\mathbf{S}$.
2. u is irrelevant for $L \cup \bar{\mathbf{S}}$ with schema $\mathcal{H}_\Sigma$.

Then, property 2. implies property 1., but not always vice versa.

Proof. 1. $\nRightarrow$ 2.: Here, a counter-example is presented. Let $L = \{\varepsilon\}$ and $\mathbf{S} = \{\varepsilon, a\}$. Then, prefix a is irrelevant for L with schema $\mathbf{S}$. However, prefix a is relevant for $L \cup \bar{\mathbf{S}} = \mathcal{H}_\Sigma \setminus \{a\}$ with schema $\mathcal{H}_\Sigma$, since for the nested words $v = \varepsilon$ and $v' = a$ we have $a \cdot v \notin L \cup \bar{\mathbf{S}}$ and $a \cdot v' \in L \cup \bar{\mathbf{S}}$.

1. $\Leftarrow$ 2.: Next, we assume property 2. Consider arbitrary nested words $v, v' \in \mathcal{N}_\Sigma$ and a nested word suffix $w \in \text{suffs}(\mathcal{N}_\Sigma)$ such that $u \cdot v \cdot w, u \cdot v' \cdot w \in \mathbf{S}$. It is sufficient to show that $u \cdot v \cdot w \in \mathbf{S}$ and $u \cdot v' \cdot w \in \mathbf{S}$ implies $u \cdot v \cdot w \in L \Leftrightarrow u \cdot v' \cdot w \in L$. This implication holds trivially if $u \cdot v \cdot w \in \bar{\mathbf{S}}$ or $u \cdot v' \cdot w \in \bar{\mathbf{S}}$. Otherwise, we have $u \cdot v \cdot w \in \mathbf{S}$ and $u \cdot v' \cdot w \in \mathbf{S}$. Property 2., as assumed, then implies $u \cdot v \cdot w \in L \Leftrightarrow u \cdot v' \cdot w \in L$. So, property 1. holds.

□

4. Hedge Automata

We recall the notion of stepwise hedge automata (SHAs), introduce their downward extension ($\text{SHA}^\downarrow$), discuss schema-completeness for both kinds of automata, define sub-hedge projection states for $\text{SHA}^\downarrow$ s, and show how to use them for evaluating $\text{SHA}^\downarrow$ s with subhedge projection. We limit ourselves to in-memory evaluation in this section but will discuss streaming evaluation in Section 9. The relation to several closely related notions of automata on hedges, forests, or nested words is discussed in some more detail in Section 12.

4.1. Stepwise Hedge Automata

SHAs are a recent notion of automata for hedges [8] that mix up bottom-up tree and left-to-right word automata in a natural manner. They extend stepwise tree automata [31] in that they operate on hedges rather than unranked trees, i.e., on sequences of letters and trees containing hedges. They differ from nested word automata (NWAs) [11,15] in that they process hedges directly rather than their linearizations to nested words.

Definition 3. A stepwise hedge automaton (SHA) is a tuple $A = (\Sigma, \mathcal{Q}, \Delta, I, F)$ where Σ and $\mathcal{Q}$ are finite sets, $I, F \subseteq \mathcal{Q}$, and $\Delta = ((a^\Delta)_{a \in \Sigma}, \langle \rangle^\Delta, @^\Delta)$, where $a^\Delta \subseteq \mathcal{Q} \times \mathcal{Q}$, $\langle \rangle^\Delta \subseteq \mathcal{Q}$, and $@^\Delta \subseteq (\mathcal{Q} \times \mathcal{Q}) \times \mathcal{Q}$. A SHA is deterministic or equivalently a dSHA if I and $\langle \rangle^\Delta$ contain at most one element, and all relations $(a^\Delta)_{a \in \Sigma}$ and $@^\Delta$ are partial functions.

The set of states $\mathcal{Q}$ subsumes a subset I of the initial state, a subset F of final states, and a subset $\langle \rangle^\Delta$ of tree initial states. The transition rules in Δ have three forms: if $(q, q') \in a^\Delta$, then we have a letter rule that we write as $q \xrightarrow{a} q'$ in Δ . If $(q, p, q') \in @^\Delta$, then we have an apply rule that we write as $q@p \rightarrow q'$ in Δ . Lastly, if $q \in \langle \rangle^\Delta$, then we have a tree initial rule that we denote as $\langle \rangle \rightarrow q$ in Δ .

In Figure 4, we illustrate the graph of a dSHA for the XPath filter `[self::list/child::item]`. The graph of SHA is obtained and drawn as usual for state automaton on words. The states are the nodes of the graph; in the example, $\mathcal{Q} = \{0, 1, 2, 3, 4, 5\}$. The transition rules define the edges. The edge of a letter rule $q \xrightarrow{a} q'$ in Δ is annotated by the letter $a \in \Sigma$. In the example, $\Sigma = \{\text{list}, \text{item}\}$. Any apply rule $q@p \rightarrow q'$ in Δ is drawn as an edge $q \xrightarrow{p} q'$ annotated by state $p \in \mathcal{Q}$. The initial state $0 \in I$ is indicated by an ingoing gray arrow, and the tree initial state $0 \in \langle \rangle^\Delta$ by an ingoing gray arrow that is annotated by symbol $\langle \rangle$. Final states, such as $4 \in F$, are indicated by a double circle.

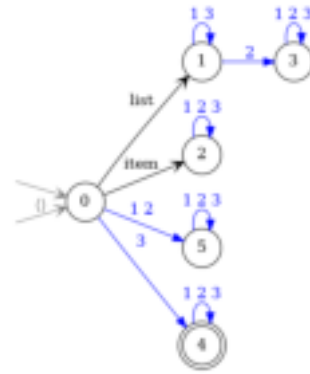


Figure 4. A dSHA for the XPath filter `[self::list/child::item]` that is schema-complete for schema `[doc]`.

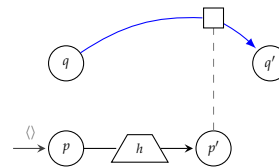
4.1.1.1. Semantics

We next define the semantics of SHAs, i.e., the transitions on hedges that it defines and the hedge language that it accepts. For any hedge $h \in \mathcal{H}_\Sigma$ we define the transition steps $q \xrightarrow{h} p$ wrt Δ such that for all $q, q', p, p' \in \mathcal{Q}$, $a \in \Sigma$, and $h, h' \in \mathcal{H}_\Sigma$

$$\frac{q \xrightarrow{a} q' \text{ in } \Delta}{q \xrightarrow{a} q' \text{ wrt } \Delta} \quad \frac{q \xrightarrow{h} q' \text{ wrt } \Delta \quad q' \xrightarrow{h'} q'' \text{ wrt } \Delta}{q \xrightarrow{h \cdot h'} q'' \text{ wrt } \Delta}$$

$$\frac{q \in \mathcal{Q}}{q \xrightarrow{\varepsilon} q \text{ wrt } \Delta} \quad \frac{\langle \rangle \rightarrow p \text{ in } \Delta \quad p \xrightarrow{h} p' \quad q @ p' \rightarrow q' \text{ in } \Delta}{q \xrightarrow{\langle h \rangle} q' \text{ wrt } \Delta}$$

The transitions can be used to evaluate hedges nondeterministically bottom-up and left-to-right using SHAs. The first three rules state how to evaluate sequences of trees and letters, such as a usual finite state automaton, while assuming that the trees were already evaluated to states. The last rule states how to evaluate a tree $\langle h \rangle$ from some state q to some state q' . This can be visualized as follows:



For any tree initial state $p \in \langle \rangle^\Delta$, one has to evaluate the subhedge h nondeterministically to some state p' . For each p' obtained, one then returns to a state q' such that $q @ p' \rightarrow q'$ in Δ nondeterministically.

A hedge is accepted by A if it permits a transition from some initial state to some final state. The language $\mathcal{L}(A)$ is the set of all accepted hedges:

$$\mathcal{L}(A) = \{h \in \mathcal{H}_\Sigma \mid q \xrightarrow{h} q' \text{ wrt } \Delta, q \in I, q' \in F\}.$$

4.1.2. Runs

Runs can represent the whole history of a single choice of the nondeterministic evaluator on a hedge. For instance, the unique successful run of the dSHA in Figure 4 on the hedge $h = \langle h' \rangle$ with subhedge $h' = \text{list} \cdot \langle \text{item} \rangle$ is the hedge $0 \cdot \langle 0 \cdot \text{list} \cdot 1 \cdot \langle 0 \cdot \text{item} \cdot 2 \rangle \cdot 3 \rangle \cdot 4$ whose computation is illustrated in Figure 5. On the top level, the run justifies the transition $0 \xrightarrow{h} 4$ wrt Δ from the initial state 0 to a final state 4. When started in state 0, the subhedge h' needs to be evaluated first, so the run on the upper level needs to be suspended until this

is done. The subrun on h' eventually justifies the transition $0 \xrightarrow{h'} 3$ wrt Δ . At this time point, the run of the upper level can be resumed and continue in state 4 since $0@3 \rightarrow 4$ in Δ .

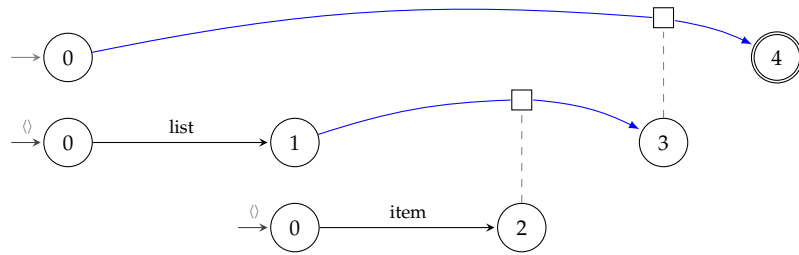


Figure 5. The unique successful run of the dSHA in Figure 4 on the hedge $\langle \text{list} \cdot \langle \text{item} \rangle \rangle$ is the hedge $0 \cdot \langle 0 \cdot \text{list} \cdot 1 \cdot \langle 0 \cdot \text{item} \cdot 2 \rangle \cdot 3 \rangle \cdot 4$ computed, as illustrated above.

When drawing runs, we illustrate any suspension/resumption mechanism with a box, for instance, the box on the edge $0 \rightarrow \square \rightarrow 4$. The box stands for a future value computed by the lower level, causing the upper level to suspend until its arrival. Eventually, the box is filled by state 3, as illustrated by $3 \dashrightarrow \square$, so that the computation can be resumed on the upper level.

We next define runs of SHAs on hedges formally. Whether a hedge with letters and states $R \in \mathcal{H}_{\Sigma \cup \mathcal{Q}}$ is a Δ -run or simple a run on a hedge $h \in \mathcal{H}_{\Sigma}$, written $R \in \text{run}^{\Delta}(h)$, is defined by the following rules:

$$\frac{q \xrightarrow{a} q' \text{ in } \Delta}{q \cdot a \cdot q' \in \text{run}^{\Delta}(a)} \quad \frac{q \cdot R \cdot q' \in \text{run}^{\Delta}(h) \quad q' \cdot R' \cdot q'' \in \text{run}^{\Delta}(h')}{q \cdot R \cdot q' \cdot R' \cdot q'' \in \text{run}^{\Delta}(h \cdot h')}$$

$$\frac{\text{true}}{q \in \text{run}^{\Delta}(\varepsilon)} \quad \frac{\langle \rangle \xrightarrow{} p \text{ in } \Delta \quad p \cdot R \cdot p' \in \text{run}^{\Delta}(h) \quad q@p' \rightarrow q' \text{ in } \Delta}{q \cdot \langle R \rangle \cdot q' \in \text{run}^{\Delta}(\langle h \rangle)}$$

Note that if $R \in \text{run}^{\Delta}(h)$, then h can be obtained by removing all states from R , i.e., $\text{proj}_{\Sigma}(R) = h$. Any run $R \in \text{run}^{\Delta}(h)$ can be identified with the mapping of prefixes of the nested word $nw(h)$ to states in $\mathcal{Q}$. Note that $nw(h) = \text{proj}_{\Sigma}(nw(R))$. For any prefix v of $nw(h)$, there exists a unique prefix r of the nested word $nw(R)$ that ends with some state $q \in \mathcal{Q}$ and satisfies $\text{proj}_{\Sigma}(r) = v$. We then call q the state of R at prefix v . The run $0 \cdot \langle 0 \cdot \text{list} \cdot 1 \cdot \langle 0 \cdot \text{item} \cdot 2 \rangle \cdot 3 \rangle \cdot 4$ on hedge $\langle \text{list} \langle \text{item} \rangle \rangle$ from Figure 5, for instance, assigns state 0 to the nested word prefixes ε and $\langle$ and state 1 to the nested word prefix $\langle \cdot \text{list}$, etc.

A Δ -run is called a run of $A = (\Sigma, \mathcal{Q}, \Delta, I, F)$ if it starts with some state in I . A run of A is successful if it ends with some state in F .

Lemma 4. $h \in \mathcal{L}(A)$ iff there exists a successful run $R \in \text{run}^{\Delta}(h)$ of A .

Proof. Straightforward with induction on h . $\square$

4.1.3. Determinization

SHAs can always be determinized using the subset construction known from the determinization of finite state automata on words or of tree automata.

Given a SHA $A = (\Sigma, \mathcal{Q}, \Delta, I, F)$, the state set of its determinization is $\mathcal{Q}^{\text{det}} = 2^{\mathcal{Q}}$. The transition rules Δ^{det} of the determinization are presented in Figure 6.

$$\begin{array}{c}
\frac{Q \subseteq \mathcal{Q} \quad a \in \Sigma \quad Q' = \{q' \in \mathcal{Q} \mid q \xrightarrow{a} q' \in \Delta, q \in Q\}}{Q \xrightarrow{a} Q' \in \Delta^{det}} \\
\frac{\langle \rangle^\Delta \neq \emptyset \quad Q_1, Q_2 \subseteq \mathcal{Q} \quad Q' = \{q' \in \mathcal{Q} \mid q_1 @ q_2 \rightarrow q' \in \Delta, q_1 \in Q_1, q_2 \in Q_2\}}{\langle \rangle^\Delta \in \Delta^{det} \quad Q_1 @ Q_2 \rightarrow Q' \in \Delta^{det}}
\end{array}$$

Figure 6. The determinization Δ^{det} of the transition rules Δ of a SHA.

4.1.4. In-Memory Membership Testing

Testing membership $h \in \mathcal{L}(A)$ for a hedge h , whose graph is stored in memory, can be carried out by computing the unique run of the determinization of A in a bottom-up and left-to-right manner, and testing whether it is successful. Alternatively, one can compute the unique set $P \subseteq \mathcal{Q}$ such that $I \xrightarrow{h} P$ with respect to the determinization of A and to test whether $P \cap F \neq \emptyset$. The computations are essentially the same, but the full history of the computation is lost when returning only the set P of states reached and not the run.

For determinizing SHAs we can rely on a variant of the usual subset construction that is well-known from finite state automata on words or standard tree automata (see e.g., [8,16]). When it comes to membership testing, on-the-fly determinization is sufficient, which creates only the part of the deterministic automaton that is used by the run on h . This part is of linear size $O(|h|)$ while the full determinization of A may be of exponential size $O(2^{|A|})$. In this way, membership $h \in \mathcal{L}(A)$ can be tested in time $O(|h||A|)$.

4.1.5. Hedge Accessibility

For any set $P \subseteq \mathcal{Q}$ we define the set of states accessible from P via a hedge as follows:

$$acc^\Delta(P) = \{q' \in \mathcal{Q} \mid \exists q \in P. \exists h \in \mathcal{H}_\Sigma. q \xrightarrow{h} q' \text{ wrt } \Delta\}.$$

We note that $acc^\Delta(P)$ can be computed from Δ and P in linear time in the size of Δ . A state is hedge accessible from P if and only if it is accessible for P in the graph of Δ . Or by computing the least fixed point of the following ground Datalog program depending on P and Δ that is of linear size in A :

$$\frac{q \in P}{acc(p).} \quad \frac{q \xrightarrow{a} q' \text{ in } \Delta}{acc(q') :- acc(q).} \quad \frac{q @ p \rightarrow q' \text{ in } \Delta}{acc(q') :- acc(q), acc(p).}$$

Clearly, $\mathcal{L}(A) = \emptyset$ if and only if $F \cap acc^\Delta(I) = \emptyset$. Therefore, emptiness can be decided in linear time for SHAs. This is in contrast to the more general notion of $SHA^\downarrow$ s that we introduce next.

4.2. Downward Stepwise Hedge Automata

The evaluation of SHAs operates in a bottom-up and left-to-right manner, but cannot pass any information top-down. We next propose an extension of SHAs to $SHA^\downarrow$ s, adding the ability to pass finite state information top-down.

$SHA^\downarrow$ s are basically equal to Neumann and Seidl's pushdown forest automata [25] but lifted from (labeled) forests to (unlabeled) hedges. See Section 12 for a more detailed discussion on the relationship of $SHA^\downarrow$ s with other automata notions and NWA's in particular.

Definition 4. A downward stepwise hedge automaton ($SHA^\downarrow$) is a tuple $A = (\Sigma, \mathcal{Q}, \Delta, I, F)$, where Σ and $\mathcal{Q}$ are finite sets, $I, F \subseteq \mathcal{Q}$, and $\Delta = ((a^\Delta)_{a \in \Sigma}, \langle \rangle^\Delta, @^\Delta)$. Furthermore, $a^\Delta \subseteq \mathcal{Q} \times \mathcal{Q}$, $\langle \rangle^\Delta \subseteq \mathcal{Q} \times \mathcal{Q}$, and $@^\Delta \subseteq (\mathcal{Q} \times \mathcal{Q}) \times \mathcal{Q}$. A $SHA^\downarrow$ is deterministic or equivalently a $dSHA^\downarrow$ if I contains at most one element, and all relations, $\langle \rangle^\Delta$, a^Δ , and $@^\Delta$, are partial functions.

The only difference compared to SHAs is the form of the tree-opening rules. If $(q, q') \in \langle \rangle^\Delta \in \mathcal{Q}$, then we have a tree initial rule that we denote as follows: $q \xrightarrow{\langle \rangle} q'$ in Δ .

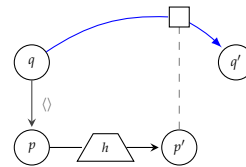
So, state q' , where the evaluation of a subhedge starts, depends on the state q of the parent.

4.2.1. Semantics

The definition of transition steps for $\text{SHA}^\downarrow$ s differs from that of SHAs only in the usage of opening rules in the following inference rule:

$$\frac{q \xrightarrow{\langle \rangle} p \text{ in } \Delta \quad p \xrightarrow{h} p' \text{ wrt } \Delta \quad q@p' \rightarrow q' \text{ in } \Delta}{q \xrightarrow{\langle h \rangle} q' \text{ wrt } \Delta}$$

This means that the evaluation of the subhedge h starts in a state p such that $q \xrightarrow{\langle \rangle} p$ in Δ .



So, the restart state p that is chosen may now depend on the state q above. This is how finite state information is passed top-down by $\text{SHA}^\downarrow$ s. SHAs, in contrast, operate purely bottom-up and left-to-right.

4.2.2. Runs

The notion of runs can be adapted straightforwardly from SHAs to $\text{SHA}^\downarrow$ s. When in state q , it is sufficient to restart the computation in subhedges on the state p , such that $q \xrightarrow{\langle \rangle} p \in \Delta$. In this way, finite state information is passed down (while for SHAs some tree initial is to be chosen that is independent of q). The only rule of runs to be changed is the following:

$$\frac{q \xrightarrow{\langle \rangle} p \text{ in } \Delta \quad p \cdot R \cdot p' \in \text{run}^\Delta(h) \quad q@p' \rightarrow q' \text{ in } \Delta}{q \cdot \langle R \rangle \cdot q' \in \text{run}^\Delta(\langle h \rangle)}$$

An example of a run of the $\text{dSHA}^\downarrow$ in Figure 7 is shown in Figure 8. Here, the information on the level of nodes is passed down. It is represented by the number of primes. For instance, when opening the first subtree labeled with item we use the transition rule $1' \xrightarrow{\langle \rangle} 0''$ stating that one moves from state 1 on the first level to state 0 on the second level.

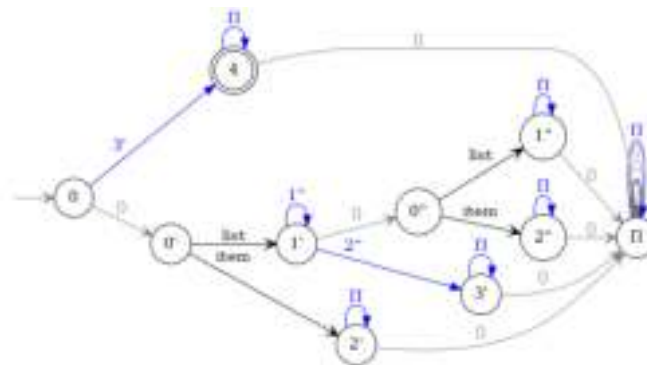


Figure 7. The deterministic $\text{SHA}^\downarrow$ with subhedge projection state Π obtained by safe-no-change projection.

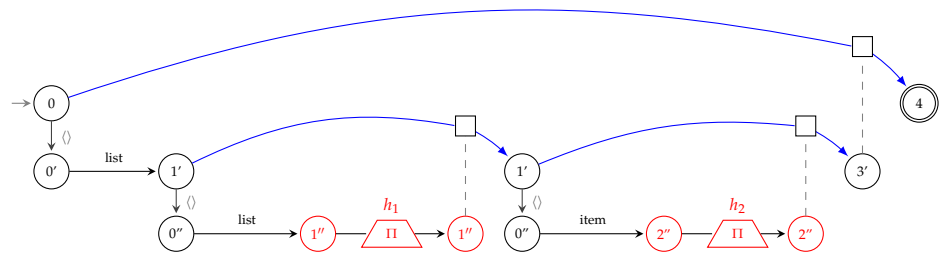


Figure 8. A successful run of the $\text{SHA}^\downarrow$ in Figure 7 on the hedge $\langle \text{list} \cdot \langle \text{list} \cdot h_1 \rangle \cdot \langle \text{item} \cdot h_2 \rangle \rangle$.

One can see that all nodes of the subtrees h_1 and h_2 are evaluated to the projection state Π . So, one can jump over these subtrees without reading them. This is justified by the fact that they are on level 2, the information that the evaluator of this $\text{SHA}^\downarrow$ was passing down.

4.2.3. In-Memory Membership Testing

Testing membership $h \in \mathcal{L}(A)$ for a $\text{SHA}^\downarrow$ s in memory can again be carried out by computing the unique run of the determinization of A . As before, the membership tester can be based on on-the-fly determinization. However, the determinization procedure for $\text{SHA}^\downarrow$ s is more complicated than for SHAs since $\text{SHA}^\downarrow$ s can pass information top-down and not only bottom-up and left-to-right. Determinization not only relies on the pure subset construction but also uses pairs of states in the subsets, basically in the same way as for nested word automata (NWA) [14,15]. This is needed to deal with the stack of states that were seen above. Therefore, each determinization step may cost time $O(|A|^5)$ as stated for instance in [23]. The upper time bound for membership testing for $\text{SHA}^\downarrow$ s is thus in time $O(|h||A|^5)$ and no longer combined linear as it is for SHAs.

4.2.4. Relationship to SHAs

Any SHA $A = (\Sigma, \mathcal{Q}, \Delta, I, F)$ can be mapped to a $\text{SHA}^\downarrow A^{\text{down}} = (\Sigma, \mathcal{Q}, \Delta^{\text{down}}, I, F)$ while preserving runs and determinism. The only change of Δ^{down} compared to Δ is described by the following rule:

$$\frac{\langle \rangle \xrightarrow{\quad} p \text{ in } \Delta \quad q \in \mathcal{Q}}{q \xrightarrow{\langle \rangle} p \text{ in } \Delta^{\text{down}}}$$

Independently of the current state $q \in \mathcal{Q}$, the $\text{SHA}^\downarrow A^{\text{down}}$ can start the evaluation of the subhedge of a subtree in any open tree state $p \in \langle \rangle^\Delta$. For instance, the $\text{dSHA}^\downarrow A^{\text{down}}$ for $\text{dSHA} A$ from Figure 4 from the introduction is provided in Figure 9. Conversely, one can convert any $\text{dSHA}^\downarrow A$ into an equivalent SHA by introducing nondeterminism to eliminate top-down processing. See Section 12.2 for the construction.

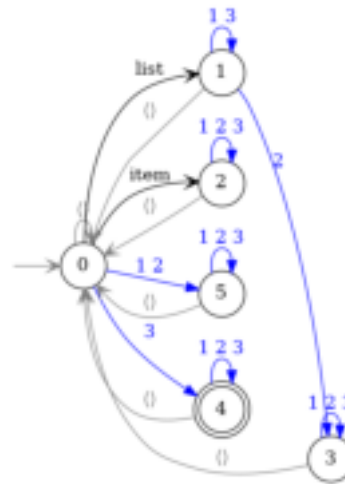


Figure 9. The $\text{SHA}^\downarrow A^{\text{down}}$ for the dSHA A for the filter $[\text{self}::\text{list}/\text{child}::\text{item}]$ in Figure 4.

4.2.5. Hedge Accessibility

Hedge accessibility can be defined for $\text{SHA}^\downarrow$ s identically as for SHAs. However, the set $\text{acc}^\Delta(P)$ can no longer be computed in linear time in the size of Δ , in contrast to SHAs. It can still be done in cubic time, similarly as for NWAs [22]. The problem is that hedge accessibility for $\text{SHA}^\downarrow$ s does no more coincide with the accessibility notion of the automaton's graph.

For instance, in the $\text{SHA}^\downarrow$ from Figure 7, we have $\text{acc}^\Delta(\{0\}) = \{4\}$. Note that $0'$ does not belong to this set, even though there is a transition rule $0 \xrightarrow{\langle \rangle} 0'$ in Δ , making node $0'$ accessible from node 0 in the graph of the automaton. This is because $0'$ is not accessible over any hedge from 0; that is, there is no hedge h such that $0 \xrightarrow{h} 0'$ wrt Δ . Note that $0 \cdot \langle \cdot \rangle \cdot 0'$ is a factor of the nested word of some run of Δ . This shows that $0'$ is accessible from 0 over a nested word prefix nevertheless. For any $\text{SHA}^\downarrow A$, hedge accessibility can be computed by the following ground Datalog program of cubic size depending on the number of states of A :

$$\frac{p@q' \rightarrow q \text{ in } \Delta \quad p \xrightarrow{\langle \rangle} p' \text{ in } \Delta}{\text{acc}(p, q) :- \text{acc}(p', q').} \quad \frac{p \in \mathcal{Q}}{\text{acc}(p, p).} \quad \frac{p \xrightarrow{a} q \text{ in } \Delta}{\text{acc}(p, q).}$$

$$\frac{p, q, r \in \mathcal{Q}}{\text{acc}(p, q) :- \text{acc}(p, r), \text{acc}(r, q).}$$

Due to the cubic time of hedge accessibility for $\text{SHA}^\downarrow$ s, we will use hedge accessibility only for SHAs where it is in linear time.

4.3. Schema Completeness

Schemas are inputs of the membership problem that we consider while completeness is a fundamental property of automata. To combine both aspects appropriately, we propose the notion of schema completeness, which we define uniformly for both SHAs and $\text{SHA}^\downarrow$ s.

We call a set of Δ transition rules $\text{SHA}^\downarrow$ complete if all its relations $(a^\Delta)_{a \in \Sigma}$, $\langle \rangle^\Delta$, and $@^\Delta$ are total. For SHAs, we require that $\langle \rangle^\Delta$ is a nonempty set (instead of a total relation). A SHA or $\text{SHA}^\downarrow A$ is called complete if Δ is complete and $I \neq \emptyset$. We can complete any SHA or $\text{SHA}^\downarrow$ by adding a fresh sink state and also transition rules into the sink state for all left-hand sides, for which no other transition rule exists. The sink state is made initial if and only if there is no initial state before. It is not final though. We denote the completion of A by $\text{compl}(A)$.

Definition 5. A partial run of A on a hedge h is a prefix r of a run of $\text{compl}(A)$ on h , such that

- r ends with some state of $\mathcal{Q}$, and

- r does not contain the sink state of $\text{compl}(A)$.

A partial run r of A on h is called *blocking* if there does not exist any partial run r' of A on h , such that r is a strict prefix of r' .

For instance, consider the hedge $h = \langle \text{list} \cdot \langle \text{item} \rangle \rangle$. The dSHA in Figure 4 has the unique run $R = 0 \cdot \langle \cdot 0 \cdot \text{list} \cdot 1 \cdot \langle \cdot 0 \cdot \text{item} \cdot 2 \rangle \cdot 3 \cdot \rangle \cdot 4$ on h that is illustrated in Figure 5. The partial runs of h are thus all prefixes of $\text{nw}(R)$ that end with some state, i.e., $0, 0 \cdot \langle \cdot 0, 0 \cdot \langle \cdot 0 \cdot \text{list} \cdot 1, \text{etc.}$ None of these partial runs are blocking.

Definition 6. A schema for an automaton A is a hedge language $S \subseteq \mathcal{H}_\Sigma$, such that $\mathcal{L}(A) \subseteq S$. We call the automaton A *schema-complete* for schema S if no hedge $h \in S$ has a blocking partial run of A .

Schemas S are usually used to restrict the type of hedges that a problem may accept as input. The dSHA pattern-matching problem for a schema S takes two inputs: a hedge $h \in S$ and an automaton A , rather than a nested regular expression e . The automaton A then selects those input hedges $h \in S$ that match the pattern, i.e., such that $h \in \mathcal{L}(A)$. For this reason, we assume that S is a schema for A , i.e., $\mathcal{L}(A) \subseteq S$. We will often assume that A is schema-complete for S , so that no partial run of A on any input hedge $h \in S$ may ever block. This can always be achieved next based on the next Lemma 5.

Example 5. The dSHA in Figure 4 for the XPath filter $[\text{self}::\text{list}/\text{child}::\text{item}]$ has signature $\Sigma = \{\text{item}, \text{list}\}$. It is schema-complete for schema $[\text{doc}]$. To make this happen, we added state 5 to this automaton, which is not used by any successful run. Still, this SHA is not complete. For completing it, we would have to run 5 into sink states by adding many transitions into it, such as $0@4 \rightarrow 5$ and $0@5 \rightarrow 5$, and also add for all other states $q \in \{1, 2, 3, 4, 5\}$ the transition rules $q \xrightarrow{\text{list}} 5, q \xrightarrow{\text{item}} 5, q@4 \rightarrow 5$, and $q@5 \rightarrow 5$.

As we will see, automaton completion would make the safe-no-change algorithm incomplete for subhedge projection for Example 5. Without schema-completeness, however, safe-no-change projection may be unsound: it might overlook the rejection of hedges inside the schema. This is why it is much better to assume only schema-completeness rather than full completeness for the input automata of safe-no-change projection. For congruence projection, schema-completeness will be assumed implicitly.

We note that schema-completeness is well-defined even for nonregular schemas. For safe-no-change projection, we indeed do not have to restrict schemas to be regular, while for congruence projection, only regular schemas are considered.

Furthermore, if A is schema-complete for the universal schema $S = \mathcal{H}_\Sigma$ and does not have inaccessible states, then A is complete.

Schema-completeness of deterministic automata for regular schemas can always be made to hold as we will show next. For this we define that A is *compatible* with schema S if for any hedge $h \in S$ there exists a run of A in $\text{run}^\Delta(h)$. Schema-completeness implies compatibility, as shown by the next lemma. The converse is true only when assuming determinism.

Lemma 5. Let A be a deterministic SHA with schema S . Then, A is compatible with S iff A is schema-complete for S .

Proof. Suppose that A is schema-complete for S . Wlog., we can assume $S \neq \emptyset$. Note that if $I = \emptyset$, then ε would be a blocking partial run for any hedge in $S \setminus \{\varepsilon\}$. Since $S \neq \emptyset$, it follows that there exists $q_0 \in I$. So, q_0 is a partial run for any hedge $h \in S \setminus \{\varepsilon\}$. Since there exist no blocking partial runs by schema-completeness, this run can be extended step by step to a run on any $h \in S$. Therefore, for any hedge $h \in S$, there exists some run by A , showing compatibility.

For the converse, let A be compatible with $\mathbf{S}$ and deterministic. We have to show that A is schema-complete for $\mathbf{S}$. Let v be a partial run of A on $h \in \mathbf{S}$. By compatibility, there exists a run $R \in \text{run}^\Delta(h)$ that starts in some initial state of A . By determinism, v must be a prefix of $nw(R)$. Thus, v is not blocking. $\square$

Proposition 1. Any SHA A with regular schema $\mathbf{S}$ can be made schema-complete for $\mathbf{S}$.

Proof. By Lemma 5, it is sufficient to make A compatible with $\mathbf{S}$. Let B be a dSHA with $\mathbf{S} = \mathcal{L}(B)$. Automaton B is compatible with $\mathbf{S}$. Since B is deterministic, Lemma 5 implies that B is schema-complete for $\mathbf{S}$. Since $\mathbf{S}$ is a schema for A with $\mathcal{L}(A) \subseteq \mathcal{L}(B)$. We then compute the completion of $\text{compl}(A)$. The product $C = B \times \text{compl}(A)$ with final states $F^C = F^B \times F^A$ is schema-complete for schema $\mathbf{S}$. Furthermore, since $\mathcal{L}(A) \subseteq \mathbf{S} = \mathcal{L}(B)$, it follows that $\mathcal{L}(C) = \mathcal{L}(B) \cap \mathcal{L}(A) = \mathcal{L}(A)$. $\square$

Finally, note that the schema $\mathbf{S}$ can be recognized by the same product $C = B \times \text{compl}(A)$ except for replacing the set of final states F^C by the set of schema final states $F^{\mathbf{S}} = F^B \times (\mathcal{Q}^A \cup \{\text{sink}\})$. We have $\mathbf{S} = \mathcal{L}(C[F^C/F^{\mathbf{S}}])$, where $C[F^C/F^{\mathbf{S}}]$ is the dSHA obtained from C by replacing the set of final states F^C with $F^{\mathbf{S}}$.

5. Subhedge Projection

We next introduce the concept of subhedge projection states for $\text{SHA}^\downarrow$ s in order to distinguish prefixes of hedges that are subhedge irrelevant, and to evaluate $\text{SHA}^\downarrow$ s with subhedge projection.

5.1. Subhedge Projection States

Intuitively, a subhedge projection state can only loop in itself until a hedge closes (and then it is applied). When going down into a subtree, the subhedge projection state may still be changed into some other looping state. When leaving the subtree, however, one must come back to the original subhedge projection state. Clearly, any sink is a subhedge projection state. The even more interesting cases, however, are subhedge projection states that are not sinks.

We start with $\text{SHA}^\downarrow$ s without any restrictions but will impose schema-completeness and determinism later on.

Definition 7. We call a state $q \in \mathcal{Q}$ a subhedge projection state of Δ if there exists $q' \in \mathcal{Q}$ called the witness of q , such that the set of transition rules of Δ containing q' or q on the leftmost position is included in the following set:

$$\begin{aligned} & \{q \xrightarrow{\langle \rangle} q', q @ q' \rightarrow q, q' \xrightarrow{\langle \rangle} q', q' @ q' \rightarrow q'\} \\ & \cup \{q' \xrightarrow{a} q', q \xrightarrow{a} q \mid a \in \Sigma\} \end{aligned}$$

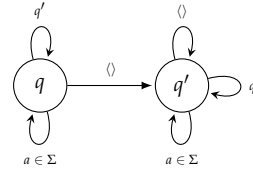
The set of all subhedge projection states of Δ is denoted by $\mathcal{Q}_{shp}^\Delta$.

For any complete $\text{SHA}^\downarrow$, the above set of transition rules must be equal to the set of all transition rules of Δ with q or q' on the leftmost position. But, if the $\text{SHA}^\downarrow$ is only schema-complete for some schema $\mathbf{S}$, then not all these transitions must be present. Note also that a subhedge projection state q may be equal to its witness q' . Therefore, any witness q' of some subhedge projection state is itself a subhedge projection state with witness q' .

In the example $\text{dSHA}^\downarrow A = (\Sigma, \mathcal{Q}, \Delta, I, F)$ in Figure 7, we have $\mathcal{Q}_{shp}^\Delta = \{\Pi, 4, 2', 3', 1'', 2''\}$. The witness of all these subhedge projection states is Π . Note that not all possible transitions are present for the states in $\mathcal{Q}_{shp}^\Delta \setminus \{\Pi\}$, given that this automaton is not complete (but still schema-complete for schema $\llbracket \text{doc} \rrbracket$).

Lemma 6. If $q \in \mathcal{Q}_{shp}^\Delta$ is a subhedge projection state and $q \xrightarrow{h} p$ wrt Δ , then $q = p$.

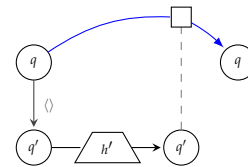
Proof. Suppose that $q \xrightarrow{h} p$ wrt Δ and that q is a subhedge projection state of Δ with witness q' . So, the only possible transitions with q or q' on the left-hand side are the following:



Suppose that $h = t_1 \cdot \dots \cdot t_n$ is a sequence of trees or letters $t_i \in \langle \mathcal{H}_\Sigma \rangle \cup \Sigma$, where $1 \leq i \leq n$. Then, there exists runs $R_i \in \text{run}^\Delta(t_i)$ and a run $R \in \text{run}^\Delta(h)$ that has the form $h = q_0 \cdot R_1 \cdot q_1 \cdot \dots \cdot R_n \cdot q_n$, where $q_0 = q$ and $q_n = p$. We prove for all $0 \leq i \leq n$ that $q_i = q$ by induction on i . For $i = 0$, this is trivial. For $i > 0$, the induction hypothesis shows that $q_{i-1} = q$.

Case $t_i = a \in \Sigma$. Then, $q_{i-1} \xrightarrow{a} q_i$ in Δ . Since $q_{i-1} = q$ is a subhedge projection state of Δ , it follows that $q_i = q$, too.

Case $t_i = \langle h' \rangle$ for some $h' \in \mathcal{H}_\Sigma$. Since $q_{i-1} = q$ is a subhedge projection state of Δ with witness q' , the subrun of R recognizing t_i must be justified by the following diagram:



So, the subrun of R of h' may only contain the witness q' and, furthermore, $q_i = q$.

□

5.2. Completeness for Subhedge Projection

Before defining when a $\text{SHA}^\downarrow$ is complete for subhedge projection, we show that subhedge projection states in runs of deterministic SHAs may only occur at irrelevant prefixes.

Proposition 2. Let $A = (\Sigma, \mathcal{Q}, \Delta, I, F)$ be a $\text{dSHA}^\downarrow$ with schema $\mathbf{S}$, $R \in \text{run}^\Delta(h)$ is the run of $\text{compl}(A)$ on some hedge $h \in \mathbf{S}$, and q is a subhedge projection state for Δ . Then, for any prefix $r \cdot q$ of $\text{nw}(R)$, the prefix $\text{proj}_\Sigma(r)$ of $\text{nw}(h)$ is subhedge irrelevant for $\mathcal{L}(A)$ for schema $\mathbf{S}$.

Proof. Let $R \in \text{run}^\Delta(h)$ be the unique run of $\text{compl}(A)$ on a hedge $h \in \mathbf{S}$. Suppose that $r \cdot q \cdot s = \text{nw}(R)$ for some subhedge projection states q of Δ and let $v = \text{proj}_\Sigma(r)$ and $w = \text{proj}_\Sigma(s)$. Since $v \cdot w = \text{nw}(h)$ and $h \in \mathbf{S}$, it follows that $v \cdot w \in \text{nw}(\mathbf{S})$. We fix some hedge $h' \in \mathcal{H}_\Sigma$, such that $v \cdot \text{nw}(h') \cdot w \in \text{nw}(\mathbf{S})$ arbitrarily. Let $\tilde{h} \in \mathcal{H}_\Sigma$, such that $\text{nw}(\tilde{h}) = v \cdot \text{nw}(h') \cdot w$. For the subhedge irrelevance of v , we must show that

$$h \in \mathcal{L}(A) \Leftrightarrow \tilde{h} \in \mathcal{L}(A).$$

“ $\Rightarrow$ ” We suppose that $h \in \mathcal{L}(A)$ and have to show that $\tilde{h} \in \mathcal{L}(A)$. For the partial run $r \cdot q$ of A on $\tilde{h}$, there must exist R' and p , such that $q \cdot R' \cdot p \in \text{run}^\Delta(h')$. Since q is a subhedge projection state of Δ , it follows by Lemma 6 that $q = p$. Hence, $r \cdot q \cdot \text{nw}(R') \cdot q \cdot s$ is a run of A on $\tilde{h}$. Since R was successful for A , s must end in F ; thus, the above run on $\tilde{h}$ is successful for A too. Hence, $\tilde{h} \in \mathcal{L}(A)$, as we have shown.

“ $\Leftarrow$ ” We suppose that $\tilde{h} \in \mathcal{L}(A)$ and have to show that $h \in \mathcal{L}(A)$. Let $\tilde{R}$ be the unique run of $\text{compl}(A)$ on $\tilde{h}$. By determinism, $\tilde{R}$ must have the prefix $r \cdot q$. So, there exist

R', p', s , such that $p \cdot R' \cdot q \in \text{run}^\Delta(h')$ and $r \cdot q \cdot R' \cdot p \cdot s = \tilde{R}$. Lemma 6 shows that $p = q$. Hence, $r \cdot q \cdot s$ is the unique run of $\text{compl}(A)$ on h and this run is accepting. So, $h \in \mathcal{L}(A)$.

□

We note that Proposition 2 would not hold without assuming determinism. To see this, we can add some sink to any dSHA as a second initial state. One can then always go to this sink, which is a subhedge projection state. Nevertheless, no prefix going to the sink may be subhedge irrelevant.

Proposition 2 shows that subhedge projection states permit distinguishing prefixes that are subhedge irrelevant for deterministic SHA[↓]s. An interesting question is whether all subhedge irrelevant prefixes can be found in this way. A closely related question is whether, for all regular patterns, there exist dSHAs that project all irrelevant subhedges.

Example 6. Reconsider Example 3 with $\Sigma = \{a\}$, a regular pattern $L = \{\langle a \rangle\}$, and a regular schema $S = L \cup \{\langle \rangle \cdot a\}$. The prefix $\langle \rangle$ is irrelevant for L and S , since all its continuations ending in the schema are rejected: there is only one such continuation, which is $\langle \rangle \cdot a$. However, a dSHA for L and S with maximal subhedge projection is given in Figure 17 but will not go into a subhedge projection state at this prefix. The reason is that it will go into the same state $\{3, 4\}D0$ for any prefix in $\langle \mathcal{N}_\Sigma \rangle$, ignoring the nested word of the irrelevant subhedge. Therefore, this state must accept $\langle a \rangle$. But, when reading an a in this state, the dSHA goes into the rejection state $\{5\}D0$, showing that the hedge from the schema $\langle \rangle \cdot a \in S$ is rejected.

The example shows that we eventually reason with sets of prefixes. We first lift the notion of subhedge irrelevance to prefix sets.

Definition 8. Let $S \subseteq \mathcal{H}_\Sigma$ be a schema and $L \subseteq S$ a language satisfying this schema. A set of nested word prefixes $U \subseteq \text{pref}(\mathcal{N}_\Sigma)$ is called subhedge relevant for L with schema S if there exist nested words $v', v'' \in \mathcal{N}_\Sigma$ prefixes $u', u'' \in U$, such that $\text{diff}_S^L(u' \cdot v', u'' \cdot v'')$. Otherwise, the set U is called subhedge irrelevant for L wrt S .

In order to explain the problem of Example 6, we define for each nested word prefix $w \in \text{pref}(\mathcal{N}_\Sigma)$ a class $\text{class}_S^L(w)$ of similar prefixes up replacing the nested word of any irrelevant subhedge by $\mathcal{N}_\Sigma$ from the left to the right. For all prefixes $w \in \text{pref}(\mathcal{N}_\Sigma)$, nested words $v \in \mathcal{N}_\Sigma$, letters $a \in \Sigma$, subsets of prefixes $U \subseteq \text{pref}(\mathcal{N}_\Sigma)$, and $l \in \hat{\Sigma}$ letters or parenthesis, the following applies:

$$\begin{aligned} \text{class}_S^L(w) &= \text{class\_pr}_{\{\varepsilon\}}(w) \\ \text{class\_pr}_U(\varepsilon) &= \begin{cases} U \cdot \mathcal{N}_\Sigma & \text{if } U \text{ irrelevant for } L \text{ wrt } S \\ U & \text{else} \end{cases} \\ \text{class\_pr}_U(w \cdot \langle \cdot v \rangle) &= \text{class\_ins}_{U'}(\langle \cdot v \rangle) \text{ where } U' = \text{class\_pr}_U(w) \\ \text{class\_pr}_U(a \cdot v) &= \text{class\_ins}_{U'}(a, v) \text{ where } U' = U \cdot a \\ \text{class\_pr}_U(\langle v' \rangle \cdot v) &= \text{class\_ins}_{U'}(\langle \cdot v \rangle) \text{ where } U' = \text{class\_pr}_U(\langle \cdot v' \rangle) \\ \text{class\_ins}_U(l, v) &= \begin{cases} U \cdot \mathcal{N}_\Sigma & \text{if } U \text{ irrelevant for } L \text{ wrt } S \\ \text{class\_pr}_{U \cdot l}(v) & \text{else} \end{cases} \end{aligned}$$

Definition 9. A prefix $u \in \text{pref}(\mathcal{N}_\Sigma)$ is strongly subhedge irrelevant for L wrt S if the set $\text{class}_S^L(u)$ is subhedge irrelevant for L wrt S .

Example 7. In our running Example 6, where $L = \{\langle a \rangle\}$, $S = L \cup \{\langle \rangle \cdot a\}$, and $\Sigma = \{a\}$, we have $\text{class}_S^L(\langle \rangle) = \langle \cdot \mathcal{N}_\Sigma$, so the prefix $\langle \rangle$ is strongly subhedge irrelevant. The set $\text{class}_S^L(\langle a \rangle) = \langle \mathcal{N}_\Sigma \rangle$ is subhedge-relevant, so the prefix $\langle a \rangle$ is not strongly subhedge irrelevant. The set $\text{class}_S^L(\langle \rangle \cdot a) = \langle \mathcal{N}_\Sigma \rangle \cdot a \cdot \mathcal{N}_\Sigma$ is subhedge irrelevant, so the prefix $\langle \rangle \cdot a$ is strongly subhedge irrelevant.

Definition 10. A $d\text{SHA}^\downarrow A = (\Sigma, \mathcal{Q}, \Delta, I, F)$ is complete for subhedge projection for schema $\mathbf{S}$ if A is schema-complete for $\mathbf{S}$ and for all hedges $h \in \mathbf{S}$ and all prefixes v of $\text{nw}(h)$ that are strongly subhedge irrelevant for Δ , the unique run of A on h goes to some subhedge projection state of Δ at v .

Example 8. The $d\text{SHA}$ in Figure 17 is complete for subhedge projection for our running Example 6 where $L = \{\langle a \rangle\}$ and $\mathbf{S} = L \cup \{\langle \rangle \cdot a\}$. It was obtained by congruence projection. On all prefixes of the subhedge irrelevant class $\text{class}_S^L(\langle \rangle)$, it goes to the subhedge projection state $\{0\}D1$. On all prefixes of the class $\text{class}_S^L(\langle a \rangle)$, which is not subhedge irrelevant, it goes to the state $\{3, 4\}D0$, which is not a subhedge projection state. On all prefixes of the subhedge irrelevant class $\text{class}_S^L(\langle \rangle \cdot a)$, it goes to the subhedge projection state $\{5\}D0$.

Example 9. The $d\text{SHA}^\downarrow$ from Figure 7 is complete for subhedge projection with schema $\llbracket \text{doc} \rrbracket$. It can be obtained by safe-no-change projection. In general, however, $d\text{SHA}^\downarrow$ s obtained by safe-no-change projection may not be complete for subhedge projection, as we will discuss in Section 6.3.

5.3. In-Memory Evaluator with Subhedge Projection

We next show how to refine the transitions for $\text{SHA}^\downarrow$ s by subhedge projection. We define the transition relation with subhedge projection $\xrightarrow{h}_{shp} \subseteq \mathcal{Q} \times \mathcal{Q}$ with respect to Δ , such that, for all hedges $h, h' \in \mathcal{H}_\Sigma$ and letters $a \in \Sigma$,

$$\begin{array}{c}
 \frac{q \in \mathcal{Q}_{shp}^\Delta \quad h \in \mathcal{H}_\Sigma}{q \xrightarrow{h}_{shp} q \text{ wrt } \Delta} \quad \frac{q \notin \mathcal{Q}_{shp}^\Delta \quad q \xrightarrow{a} q' \text{ in } \Delta}{q \xrightarrow{a}_{shp} q' \text{ wrt } \Delta} \quad \frac{q \notin \mathcal{Q}_{shp}^\Delta}{q \xrightarrow{\varepsilon}_{shp} q \text{ wrt } \Delta} \\
 \\
 \frac{q \notin \mathcal{Q}_{shp}^\Delta \quad q \xrightarrow{h}_{shp} q' \text{ wrt } \Delta \quad q' \xrightarrow{h'}_{shp} q'' \text{ wrt } \Delta}{q \xrightarrow{h \cdot h'}_{shp} q'' \text{ wrt } \Delta} \\
 \\
 \frac{q \notin \mathcal{Q}_{shp}^\Delta \quad q \xrightarrow{\langle \rangle} p \text{ in } \Delta \quad p \xrightarrow{h}_{shp} p' \quad q @ p' \rightarrow q' \text{ in } \Delta}{q \xrightarrow{\langle h \rangle}_{shp} q' \text{ wrt } \Delta}
 \end{array}$$

The first rule says that subhedge projecting transitions stay in subhedge projection states until the end of the current subhedge is reached. This is correct by Lemma 6 under the condition that there are no blocking runs. The other rules state that the evaluator behaves without subhedge projection in other states.

Proposition 3. Let $A = (\Sigma, \mathcal{Q}, \Delta, I, F)$ be a $\text{SHA}^\downarrow$ that is schema-complete for $\mathbf{S}$. Then, for all hedges $h \in \mathbf{S}$ and states $q, p \in \mathcal{Q}$, the following applies:

$$q \xrightarrow{h} p \text{ wrt } \Delta \text{ iff } q \xrightarrow{h}_{shp} p \text{ wrt } \Delta$$

Proof. We distinguish two cases as follows:

Case $q \notin \mathcal{Q}_{shp}^\Delta$. Then for any $p \in \mathcal{Q}$, $q \xrightarrow{h} p \text{ wrt } \Delta$ iff $q \xrightarrow{h}_{shp} p \text{ wrt } \Delta$ by definition of the projecting transitions.

Case $q \in \mathcal{Q}_{shp}^\Delta$. Then, $q \xrightarrow{h}_{shp} q \text{ wrt } \Delta$. Since $h \in \mathbf{S}$ and A are schema-complete for $\mathbf{S}$, there exists some run of A on h ; thus, the state is q , such that $q \xrightarrow{h} q' \text{ wrt } \Delta$. Lemma 6 shows that $q' = q$, thus $q \xrightarrow{h} q \text{ wrt } \Delta$.

□

For evaluating a nondeterministic $\text{SHA}^\downarrow A = (\Sigma, \mathcal{Q}, \Delta, I, F)$ with subhedge projection on some hedge h whose graph is stored in memory, we can compute the unique subset $P \subseteq \mathcal{Q}$, such that $I \xrightarrow{h}_{shp} P$ with respect to the transition relation Δ^{det} of determinization

of A and test whether $P \cap F \neq \emptyset$. When reaching any subhedge projection state P' while doing so, the evaluator can directly jump to the end of the current subhedge by applying the “stay” rule of the determinization of A :

$$\frac{P' \in \mathcal{Q}_{shp}^{\Delta^{det}} \quad h \in \mathcal{H}_{\Sigma}}{P' \xrightarrow{h}_{shp} P' \text{ wrt } \Delta^{det}}$$

Thereby, it will avoid visiting any subhedge in a subhedge projection state of the determinization of A . The running time will thus depend linearly on the size of the non-projected part of h under the assumption that the determinization of A is available.

As before, only the needed part of the determinization of A for evaluating h is to be produced on-the-fly. Overall, this costs polynomial time in the size of the needed part of the determinization (but not necessarily linear time since the determinization of $\text{SHA}^{\downarrow}$ s is more costly than for SHAs). Also note that whenever discovering a new state P' of the determinization of A , we have to test whether it is a subhedge projection state. For this, we must compute the state P'' , such that $P' \xrightarrow{\langle \rangle} P'' \text{ wrt } \Delta^{det}$ and test whether only the transition rules allowed for subhedge projection states for P' with witness P'' are available wrt Δ^{det} .

6. Safe-No-Change Projection

We want to solve the regular pattern-matching problem for hedges with subhedge projection. We assume that the regular pattern is given by a dSHA while the schema may be arbitrary, except that the dSHA must be schema-complete (see Section 4.3). We then present the safe-no-change projection algorithm, which compiles the dSHA into some $\text{dSHA}^{\downarrow}$ while introducing subhedge projection states.

The idea of the safe-no-change projection is to push information top-down by which to distinguish looping states that will safely no more be changed. Thereby, subhedge projection states are produced, as we will illustrate by examples. The soundness proof of safe-no-change projection is nontrivial and instructive for the soundness proof of congruence projection in Section 7.3. Completeness for subhedge projection cannot be expected, since the safe-no-change projection does only take simple loops into account. More general loops cannot be treated and it is not clear how that could be done. The congruence projection algorithm in Section 7.2 will eventually give an answer to this.

We keep the safe-no-change compiler independent of the schema and, therefore, do not have to assume its regularity. But, we have to assume the schema-completeness of the input dSHA in order to show that the output $\text{dSHA}^{\downarrow}$ recognizes the same language within the schema. The regular pattern-matching problem can then be solved in memory and with subhedge projection by evaluating the $\text{dSHA}^{\downarrow}$ on the input hedge in memory and with subhedge projection, as discussed in Section 5.3.

6.1. Algorithm

We now describe the safe-no-change projection algorithm. Let $A = (\Sigma, \mathcal{Q}, \Delta, I, F)$ be a SHA with schema $\mathbf{S}$. The algorithm takes the dSHA A as input and compiles it to some $\text{dSHA}^{\downarrow} A^{snc}$. We will state why A^{snc} is sound for subhedge projection under the condition that A is schema-complete for schema $\mathbf{S}$. We do *not* even have to assume that the schema $\mathbf{S}$ is regular. The proof is eight pages long and, therefore, delegated to the appendix.

For any set $P \subseteq \mathcal{Q}$, we define the set of states that safely lead to P :

$$\text{safe}^{\Delta}(P) = \{q \in \mathcal{Q} \mid \text{acc}^{\Delta}(\{q\}) \subseteq P\}.$$

So, state q is safe for P if any hedge read from q on which the run with Δ does not block must reach some state in P . We note that $\text{safe}^{\Delta}(P)$ can be computed in linear time in the

size of Δ using inverse hedge accessibility from P . We define the set of states that will no longer change:

$$no-change^\Delta = \{q \mid q \in safe^\Delta(\{q\})\}.$$

So, $q \in no-change^\Delta$ if and only if $acc^\Delta(\{q\}) \subseteq \{q\}$. This means that all transition rules starting in q must loop in q . In the example automaton from Figure 4, the self-looping states are those in $no-change^\Delta = \{2, 3, 4, 5\}$.

Lemma 7. Let $A = (\Sigma, \mathcal{Q}, \Delta, I, F)$ be a SHA that is schema-complete for schema S . For any hedge $h \in S$ and state $q \in I \cap no-change^\Delta$, it then holds that $q \xrightarrow{h} q$ wrt Δ .

Proof. The schema-completeness of A for S applied to $h \in S$ and $q \in I$ yields the existence of some state $q' \in \mathcal{Q}$, such that $q \xrightarrow{h} q'$ wrt Δ . Let q' be any such state. Note that $q' \in acc^\Delta(q)$. Since $q \in no-change^\Delta$, we have $q \in safe^\Delta(\{q\})$, so that $q' \in acc^\Delta(\{q\})$ implies $q' = q$. This proves $q \xrightarrow{h} q$ wrt Δ . $\square$

For any state $q \in \mathcal{Q}$ and subset of states $Q \subseteq \mathcal{Q}$, we define the following:

$$\begin{aligned} s-down^\Delta(q, Q) &= safe^\Delta(\{p \in \mathcal{Q} \mid q @^\Delta p \subseteq Q\}) \\ s-no-change^\Delta(q) &= s-down^\Delta(q, \{q\}) \end{aligned}$$

A state belongs to $s-down^\Delta(q, Q)$ if all states $p \in acc^\Delta(q)$ satisfy $q @^\Delta p \subseteq Q$. So, p is a state in a down level that will safely go up to some state in Q . A state p belongs to $s-no-change^\Delta(q)$ if it safely does not change q , i.e., if $\{q\} @^\Delta acc^\Delta(\{p\}) \subseteq \{q\}$.

We next compile the SHA A to a $SHA^\downarrow A^{snc} = (\Sigma, \mathcal{Q}^{snc}, \Delta^{snc}, I^{snc}, F^{snc})$. For this, let Π be a fresh symbol and consider the state set as follows:

$$\mathcal{Q}^{snc} = \{\Pi\} \uplus (\mathcal{Q} \times 2^\mathcal{Q}).$$

In practice, we restrict the state space to the those states that are accessible or clean the dSHAs keeping only those states that are used in some successful run. But, in the worst case, the construction may indeed be exponential. Example 15 can be adapted to show this.

A pair (q, P) means the state q will safely no longer change in the current subhedge if $q \in P \cup no-change^\Delta$. In this case, the subhedge can be projected. The sets of initial and final states are defined as follows:

$$I^{snc} = I \times \{\emptyset\} \quad F^{snc} = F \times \{\emptyset\}.$$

How to generate the transition rules of A^{snc} from those of A is described in Figure 10. On states assigned on top-level (q, P) , the set P is empty so that only the states in $no-change^\Delta$ are safe for no change. This is why the definition of I^{snc} and F^{snc} use $P = \emptyset$. When going down from a state (q, P) for which q is safe to not change, i.e., $q \in P \cup no-change^\Delta$, then the following rule is applied:

$$\frac{q \in P \cup no-change^\Delta}{(q, P) \xrightarrow{\langle \rangle} \Pi \text{ in } \Delta^{snc}}.$$

The evaluation on the lower level goes to the extra state Π , where it then loops until going back to q on the upper level. When going down from some state (q, P) , such that $q \notin P \cup no-change^\Delta$, then the following rule is applied:

$$\frac{\langle \rangle \xrightarrow{\langle \rangle} q' \text{ in } \Delta \quad q \notin P \cup no-change^\Delta}{(q, P) \xrightarrow{\langle \rangle} (q', s-no-change^\Delta(q)) \text{ in } \Delta^{snc}}$$

The states in the set $s\text{-no-change}^\Delta(q)$ on the lower level safely will not make q change on the upper level for any subhedge to come (we could detect more irrelevant subhedges by allowing q to change but only in a unique manner. This can be obtained with the alternative definition: $s\text{-no-change}'^\Delta(q) = \cup_{r \in \mathcal{Q}} s\text{-down}^\Delta(q, \{r\})$. Computing this set would require quadratic time $O(n |A|)$, while computing $s\text{-no-change}^\Delta(q)$ can be carried out in linear time $O(|A|)$.

$$\begin{array}{c}
\frac{q \xrightarrow{a} q' \text{ in } \Delta \quad q \notin P \cup \text{no-change}^\Delta}{(q, P) \xrightarrow{a} (q', P) \text{ in } \Delta^{snc}} \quad \frac{a \in \Sigma \quad q \in P \cup \text{no-change}^\Delta}{(q, P) \xrightarrow{a} (q, P) \text{ in } \Delta^{snc}} \\
\\
\frac{\langle \rangle \xrightarrow{\langle \rangle} q' \text{ in } \Delta \quad q \notin P \cup \text{no-change}^\Delta}{(q, P) \xrightarrow{\langle \rangle} (q', s\text{-no-change}^\Delta(q)) \text{ in } \Delta^{snc}} \quad \frac{q \in P \cup \text{no-change}^\Delta}{(q, P) \xrightarrow{\langle \rangle} \Pi \text{ in } \Delta^{snc}} \\
\\
\frac{q@p \rightarrow q' \text{ in } \Delta \quad q \notin P \cup \text{no-change}^\Delta}{(q, P)@(p, s\text{-no-change}^\Delta(q)) \rightarrow (q', P) \text{ in } \Delta^{snc}} \quad \frac{q \in P \cup \text{no-change}^\Delta}{(q, P)@\Pi \rightarrow (q, P) \text{ in } \Delta^{snc}} \\
\\
\frac{a \in \Sigma}{\Pi \xrightarrow{a} \Pi \text{ in } \Delta^{snc}} \quad \frac{\text{true}}{\Pi@\Pi \rightarrow \Pi \text{ in } \Delta^{snc}} \quad \frac{\text{true}}{\Pi \xrightarrow{\langle \rangle} \Pi \text{ in } \Delta^{snc}}
\end{array}$$

Figure 10. The transition rules of the $\text{SHA}^\downarrow A^{snc}$ inferred from those of the $\text{SHA} A$.

When applied to the SHA in Figure 4 for $[\text{self}::\text{list}/\text{child}::\text{item}]$, the construction yields the $\text{SHA}^\downarrow$ in Figure 11, which is indeed equal to the $\text{SHA}^\downarrow$ from Figure 7 up to state renaming. When run on the hedge $\langle \text{list} \cdot \langle \text{list} \cdot h_1 \rangle \cdot \langle \text{item} \cdot h_2 \rangle \rangle$, as shown in Figure 8, it does not have to visit subhedges h_1 or h_2 since all of them will be reached starting from the projection state Π .

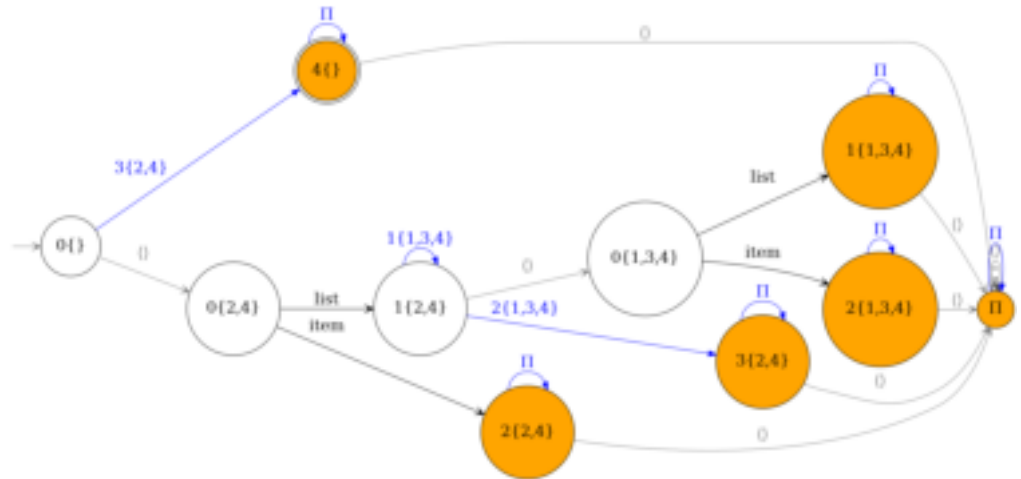


Figure 11. The safe-no-change projection $\text{dSHA}^\downarrow A^{snc}$ constructed from the $\text{dSHA} A$ for query $[\text{self}::\text{list}/\text{child}::\text{item}]$ is shown in Figure 4. Subhedge projection states are colored in orange. Useless states and transitions leading out of schema $\llbracket \text{doc} \rrbracket$ are omitted. State Π has a transition labeled by wildcard letter “—”, which stands for either letter `list` or `item`. This $\text{dSHA}^\downarrow$ is equal to the $\text{dSHA}^\downarrow$ in Figure 7 up to the state renaming $0 = (0, \{\})$, $0' = (0, \{2, 4\})$, $0'' = (0, \{1, 3, 4\})$, $1' = (1, \{2, 4\})$, $1'' = (1, \{1, 3, 4\})$, $2' = (2, \{2, 4\})$, $2'' = (2, \{1, 3, 4\})$, $3' = (3, \{2, 4\})$, and $4 = (4, \{\})$. Recall that $\text{no-change}^\Delta = \{2, 3, 4, 5\}$.

6.2. Soundness

We next state and prove a soundness result for safe-no-change projection.

Theorem 1 (Soundness of safe-no-change projection.). *If a SHA A is schema-complete for some schema S , then safe-no-change projection for A preserves the language within this schema: $\mathcal{L}(A^{snc}) \cap S = \mathcal{L}(A)$.*

The projecting in-memory evaluator of A^{snc} will be more efficient than the non-projecting evaluator of A . Note that the size of A^{snc} may be exponentially bigger than that of A . Therefore, for evaluating a dSHA A with subhedge projection on a given hedge h , we may prefer only the needed part of A^{snc} on-the-fly. This part has a size of $O(|h|)$ and can be computed in time $O(|A| |h|)$ so the exponential preprocessing time is avoided.

6.3. Incompleteness

Safe-no-change projection may be incomplete for subhedge projection so that not all prefixes that are strongly subhedge irrelevant are mapped to subhedge projection states. This is shown by the counter example dSHA A for the XPath filter $[child::item]$ in Figure 12. Its safe-no-change projection A^{snc} is given in Figure 13. Note that the prefix $u = \langle item.\langle item \text{ has the class } class_S^{\mathcal{L}(A)}(u) = \langle item.\langle item.\mathcal{N}_\Sigma$, which is subhedge irrelevant so u is strongly subhedge irrelevant for the XPath filter $[child::item]$. Nevertheless, it leads to the state $(2, \{1, 3, 5, 6\})$, which is not a subhedge projection state since $2 \notin \{1, 3, 5, 6\}$. The problem is that this state can still be changed to $(4, \{1, 3, 5, 6\})$. This state is somehow equivalent with respect to the filter but not equal to $(2, \{1, 3, 5, 6\})$.

Another incompleteness problem should be mentioned: safe-no-change projection is sensitive to automata completion as noticed earlier in Example 5. This is because a state may belong to $no-change^\Delta$ before completion but no more after adding a sink. Nevertheless, such states never change on any tree satisfying the schema. This problem can be applied, for instance, to example dSHA in Figure 4 for XPath query $[self::list/child::item]$. Therefore, it was important to assume only schema-completeness for safe-no-change projection and not to impose full completeness.

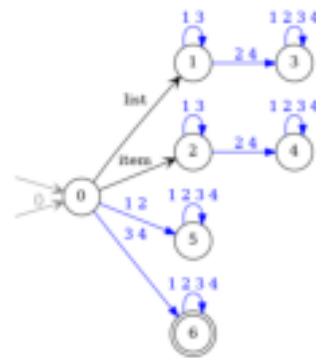


Figure 12. A schema-complete dSHA for the XPath filter $[child::item]$. It is a counter example for the completeness of safe-no-change projection, see Figure 13.

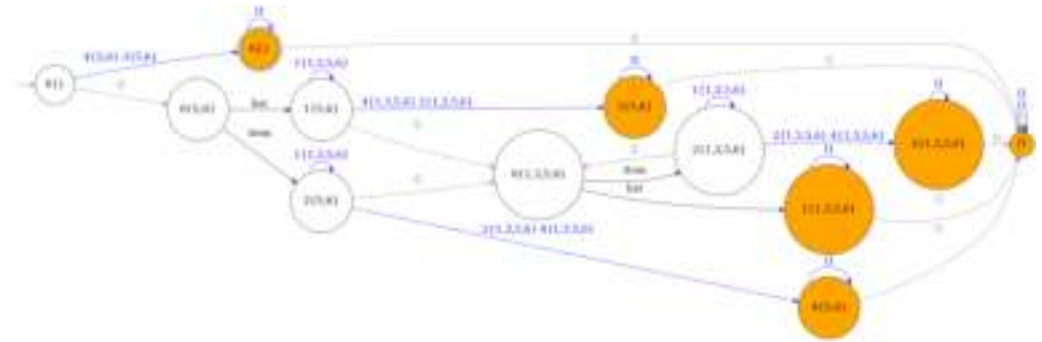


Figure 13. The safe-no-change projection A^{SNC} of the dSHA A in Figure 12. It is incomplete for subhedger projection with schema $\llbracket doc \rrbracket$ at the state $(2, \{1, 3, 5, 6\})$: this state is not a subhedger projection state, even though the prefixes $\langle list \cdot \langle item$ and $\langle item \cdot \langle item$ leading to it are subhedger irrelevant.

7. Congruence Projection

We present the congruence projection algorithm for detecting irrelevant subhedges for regular hedge patterns with regular schema restrictions. We prove that congruence projection is complete for subhedger projection, resolving the incompleteness of safe-no-change projection. For this, congruence projection may no more ignore the schema, so we have to assume that the input schema is regular too.

7.1. Ideas

The starting idea of congruence projection is to resolve the counter example for the completeness of safe-no-change projection in Figure 13. There, a state is changed when reading an irrelevant subhedger hedge. But, the state change moves to a somehow equivalent state, so it is not really relevant. Therefore, the idea is to detect when a state always remains equivalent rather than unchanged when reading an arbitrary subhedger. This is eventually done by the congruence projection in Figure 14 that we are going to construct.

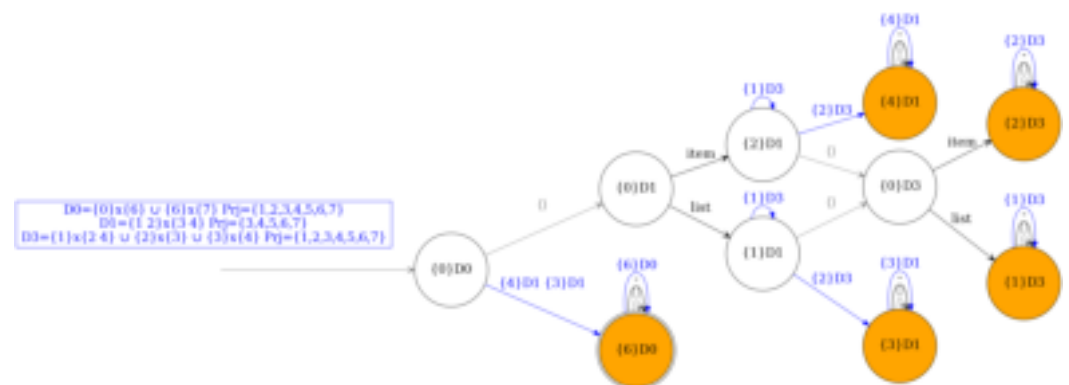


Figure 14. The congruence projection $dSHA^L A^{Cgr}(\llbracket doc \rrbracket)$ for the counter example of the completeness of safe-no-change projection, i.e., the dSHA A for the filter $[child::item]$ in Figure 12 with the schema final states $F_S = \{0, 5, 6\}$. Note that state $\{1\}D3$ that is reached over the prefix $\langle list \cdot \langle item$ and the state $\{2\}D3$ that is reached over the prefix $\langle item \cdot \langle item$ are subhedger irrelevant, and thus, subhedger projection states.

The obvious question is as follows: which equivalence relation do we choose? Suppose that we want to test whether a hedge satisfying a regular schema S matches a regular pattern L . In the restricted case of words without schema restrictions, the idea could be to use Myhill–Nerode’s congruence $\overline{diff}_{\Sigma^*}^L$ but mapped to states. In the general case, however, the situation becomes more complex, given that in the general case $\overline{diff}_S^L$ may fail to be an

equivalence relation. So, it may not be a congruence, as already illustrated in Example 4. Furthermore, the treatment of the nesting of hedges will require to update the considered relation when moving down a hedge.

7.1.1. Difference Relations on States

The congruence projection algorithm will maintain a difference relation on states of the dSHA which at the beginning will be induced by the difference relation on hedges diff_S^L and updated whenever moving down a hedge.

Definition 11. Let $(\Sigma, \mathcal{Q}, \Delta, \_ , \_)$ be a dSHA. A difference relation for Δ is a symmetric relation on states $D \subseteq \mathcal{Q} \times \mathcal{Q}$, such that for all $h \in \mathcal{H}_\Sigma$,

$$(q \xrightarrow{h} q' \text{ wrt } \Delta \wedge p \xrightarrow{h} p' \text{ wrt } \Delta \wedge (q', p') \in D) \Rightarrow (q, p) \in D.$$

The set of all difference relations for Δ is denoted by $\mathcal{D}^\Delta$.

We call a subset $Q \subseteq \mathcal{Q}$ compatible with a difference relation $D \in \mathcal{D}^\Delta$ if $Q^2 \cap D = \emptyset$. This means that no two states of Q may be different with respect to D .

Definition 12. Let $(\Sigma, \mathcal{Q}, \Delta, \_ , \_)$ be a SHA and D a difference relation for Δ . A subset of states $Q \subseteq \mathcal{Q}$ is subhedge irrelevant for D wrt Δ if $\text{acc}^\Delta(Q)$ is compatible with D . The set of all subhedge irrelevant subsets of states for D wrt. Δ thus is as follows:

$$\text{dPrj}^\Delta(D) = \{Q \subseteq \mathcal{Q} \mid \text{acc}^\Delta(Q)^2 \cap D = \emptyset\}.$$

A state $q \in \mathcal{Q}$ is subhedge irrelevant for D if the singleton $\{q\}$ is subhedge irrelevant for D . The set of all subhedge irrelevant states of $\mathcal{Q}$ is denoted by $\text{Prj}^\Delta(D)$.

We consider subhedge-irrelevance for subsets of states since the congruence projection algorithm has to eliminate the nondeterminism that it introduces by a kind of SHA determinization. Determinization is necessary in order to recognize subhedge irrelevant prefixes properly: all single states in a subset may be subhedge irrelevant while the whole subset is not.

7.1.2. Least Difference Relations

For any binary relation $R \subseteq \mathcal{Q} \times \mathcal{Q}$, let $\text{ldr}^\Delta(R)$ be the least difference relation on states that contains R .

Lemma 8. $(p_1, p_2) \in \text{ldr}^\Delta(R) \Leftrightarrow \exists (q_1, q_2) \in R. \exists h \in \mathcal{H}_\Sigma. p_1 \xrightarrow{h} q_1 \text{ wrt } \Delta \wedge p_2 \xrightarrow{h} q_2 \text{ wrt } \Delta$.

Proof. The set $\{(p_1, p_2) \in \mathcal{Q}^2 \mid \exists (q_1, q_2) \in R. \exists h \in \mathcal{H}_\Sigma. p_1 \xrightarrow{h} q_1 \text{ wrt } \Delta \wedge p_2 \xrightarrow{h} q_2 \text{ wrt } \Delta\}$ clearly is a difference relation that contains R , and thus, contains the least such difference relation $\text{ldr}^\Delta(R)$. Conversely, each pair in the above set must be contained in any difference relation containing R , and thus, in $\text{ldr}^\Delta(R)$. $\square$

Lemma 9. For any $R \subseteq \mathcal{Q} \times \mathcal{Q}$, the difference relation $\text{ldr}^\Delta(R)$ is the value of predicate D in the least fixed point of the ground datalog program generated by the following inference rules:

$$\begin{array}{c} \frac{p, q \in \mathcal{Q}}{D(p, q) :- R(p, q).} \quad \frac{p_1 \xrightarrow{a} p_2 \text{ wrt } \Delta \quad q_1 \xrightarrow{a} q_2 \text{ wrt } \Delta}{D(p_1, q_1) :- D(p_2, q_2).} \\[10pt] \frac{p_1 @ p \rightarrow p_2 \text{ wrt } \Delta \quad q_1 @ q \rightarrow q_2 \text{ wrt } \Delta}{D(p_1, q_1) :- D(p_2, q_2).} \quad \frac{p, q \in \mathcal{Q}}{D(q, p) :- D(p, q).} \end{array}$$

Proof. The first inference rule guarantees that $R \subseteq D$. The three later inference rules characterize difference relations $D \in \mathcal{D}^\Delta$. The second and third rules state that differences in D are propagated backwards over hedges h . This is done recursively by treating letter hedges $h = a$ by the second rule and tree hedges $h = \langle h' \rangle$ by the third rule. The fourth rule expresses the symmetry of difference relations D . So, the least fixed point of the datalog program generated by the above rules contains R and is the least relation that satisfies all axioms of difference relations, so it is $ldr^\Delta(R)$. $\square$

Proposition 4. For any $R \subseteq \mathcal{Q} \times \mathcal{Q}$, the least difference relation $ldr^\Delta(R)$ can be computed in time $O(|A|^2)$.

Proof. The size of the ground datalog program computing $ldr^\Delta(R)$ from Lemma 9 is at most quadratic in the size of A , so its least fixed point can be computed in time $O(|A|^2)$. $\square$

7.2. Algorithm

We now define the congruence projection algorithm by a compiler from dSHAs and a set of schema final states to $\text{SHA}^\downarrow$ s, which, when run on a hedge, can detect all subhedge irrelevant prefixes.

7.2.1. Inputs and Outputs

As inputs, the congruence projection algorithm is given a dSHA $A = (\Sigma, \mathcal{Q}, \Delta, I, F)$ for the regular pattern and a set F_S with $F \subseteq F_S \subseteq \mathcal{Q}$. The dSHA defines the regular language $L = \mathcal{L}(A)$ while the regular schema S is recognized by the dSHA $A[F/F_S] = (\Sigma, \mathcal{Q}, \Delta, I, F_S)$. So, the same base automaton defines the regular language and regular schema by choosing different sets of final states.

Example 10. In the example dSHA in Figure 4 for the XPath filter `[self::list/child::item]` with schema `[[doc]]`, we have $F = \{4\}$ and $F_S = \{4, 5\}$. In the automaton for the counter example `[child::item]` for safe-no-change projection in Figure 12, we have $F = \{6\}$ and $F_S = \{5, 6\}$.

We note that, if L and S were given by independent SHAs, then we can obtain a common dSHA A and a set F_S as above by computing the product of the dSHAs for L and S and completing it. As noticed in [16], it may be more efficient to determinize the SHA for S in a first step, build the product of the SHA for L with the dSHA for S in a second, and determinize this product in the third step.

The congruence projection of A wrt. F_S will maintain in its current configuration a subset of states $Q \subseteq \mathcal{Q}$ and a difference relation on states in $D \in \mathcal{D}^\Delta$. Thereby, the congruence projection $\text{dSHA}^\downarrow$ can always detect whether the current prefix is subhedge irrelevant for L wrt. schema S by testing whether the current set of states Q is subhedge irrelevant for the current difference relation D .

7.2.2. Initial Difference Relation

The initial difference relation on states $D_{init}^{A, F_S} \in \mathcal{D}^\Delta$ is induced from the difference relation on prefixes diff_S^L as follows:

$$D_{init}^{A, F_S} = \{(q', q'') \mid \exists (v', v'') \in \text{diff}_S^L. \exists q_0 \in I. q_0 \xrightarrow{\text{hdg}(v')} q' \text{ wrt } \Delta \wedge q_0 \xrightarrow{\text{hdg}(v'')} q'' \text{ wrt } \Delta\}.$$

The next lemma indicates how D_{init}^{A, F_S} can be defined from A and F_S without reference to the languages $L = \mathcal{L}(A)$ and $S = \mathcal{L}(A[F/F_S])$.

Lemma 10. $D_{init}^{A, F_S} = ldr^\Delta(F \times (F_S \setminus F)) \cap \text{acc}^\Delta(I)^2$.

Proof. For one direction let $(p', p'') \in D_{init}^{A, F_S}$. Then, there exist nested words $(v', v'') \in \text{diff}_S^L$ and an initial state $q_0 \in I$, such that $q_0 \xrightarrow{\text{hdg}(v')} p'$ wrt Δ and $q_0 \xrightarrow{\text{hdg}(v'')} p''$ wrt Δ . Since v', v'' are nested words, hedge accessibility $(p', p'') \in \text{acc}^\Delta(I)^2$ follows. Furthermore, $(v', v'') \in \text{diff}_S^L$ requires the existence of a hedge $h \in \mathcal{H}_\Sigma$, such that $\text{hdg}(v') \cdot h \in L$ and $\text{hdg}(v'') \cdot h \in S \setminus L$. Hence, there are states $q' \in F$ and $q'' \in F_S \setminus F$, such that $p' \xrightarrow{h} q'$ and $p'' \xrightarrow{h} q''$. Using Lemma 8, this implies that $(p', p'') \in \text{ldr}^\Delta(F \times (F_S \setminus F))$.

For the other direction, let $(p', p'') \in \text{ldr}^\Delta(F \times (F_S \setminus F)) \cap \text{acc}^\Delta(I)^2$. Using Lemma 8, property $(p', p'') \in \text{ldr}^\Delta(F \times (F_S \setminus F))$ shows that there exist states $q' \in F$, $q'' \in F_S \setminus F$, and $h \in \mathcal{H}_\Sigma$, such that $p' \xrightarrow{h} q'$ wrt Δ and $p'' \xrightarrow{h} q''$ wrt Δ . From $(p', p'') \in \text{acc}^\Delta(I)^2$, it follows that there are nested words $v', v'' \in \mathcal{N}_\Sigma$ and an initial state $q_0 \in I$, such that $q_0 \xrightarrow{\text{hdg}(v')} p'$ wrt Δ and $q_0 \xrightarrow{\text{hdg}(v'')} p''$ wrt Δ . Hence, $(v', v'') \in \text{diff}_S^L$, so that $(q', q'') \in D_{init}^{A, F_S}$. $\square$

As a consequence of Lemma 10 and Proposition 4, the initial difference relation D_{init}^{A, F_S} can be computed in time $O(|A|^2)$ from A and F_S .

Proposition 5 (Soundness of the initial difference relation). *Let $A = (\Sigma, Q, \Delta, I, F)$ be a complete dSHA, $F \subseteq F_S \subseteq Q$, $L = \mathcal{L}(A)$, and $S = \mathcal{L}(A[F/F_S])$. For any hedge $h \in \mathcal{H}_\Sigma$ and state $q \in Q$, such that $q_0 \xrightarrow{h} q$ wrt Δ for some $q_0 \in I$, the nested word $nw(h)$ is subhedge irrelevant for L and S if and only if q is subhedge irrelevant for D_{init}^{A, F_S} wrt. Δ .*

Proof. We show that $nw(h)$ is subhedge relevant for L and S if and only if q is subhedge relevant for D_{init}^{A, F_S} wrt. Δ .

“ $\Rightarrow$ ” Let $nw(h)$ be subhedge relevant for L and S . Then, there exist hedges $h', h'' \in \mathcal{H}_\Sigma$ and a nested word $w \in \mathcal{N}_\Sigma$, such that

$$nw(h) \cdot nw(h') \cdot w \in L \wedge nw(h) \cdot nw(h'') \cdot w \in S \setminus L.$$

Since A is deterministic and $q_0 \xrightarrow{h} q$ wrt Δ for the unique $q_0 \in I$, it follows that there exist states $p', p'' \in Q$, $q' \in F$, and $q'' \in F_S \setminus F$, such that

$$q \xrightarrow{h'} p' \xrightarrow{\text{hdg}(w)} q' \text{ wrt } \Delta \wedge q \xrightarrow{h''} p'' \xrightarrow{\text{hdg}(w)} q'' \text{ wrt } \Delta.$$

From $q' \in F$ and $q'' \in F_S \setminus F$, it follows that $(q', q'') \in D_{init}^{A, F_S}$. Since D_{init}^{A, F_S} is a difference relation, $p' \xrightarrow{\text{hdg}(w)} q'$ wrt Δ and $p'' \xrightarrow{\text{hdg}(w)} q''$ wrt Δ , this implies that $(p', p'') \in D_{init}^{A, F_S}$, too. Hence, $(p', p'') \in \text{acc}^\Delta(\{q\})^2 \cap D_{init}^{A, F_S}$, i.e., q is relevant for D_{init}^{A, F_S} wrt. Δ .

“ $\Leftarrow$ ” Let q be subhedge relevant for D_{init}^{A, F_S} wrt. Δ . Then, there exist a pair $(p', p'') \in \text{acc}^\Delta(\{q\})^2 \cap D_{init}^{A, F_S}$. So, there are hedges h' and h'' , such that

$$q \xrightarrow{h'} p' \text{ wrt } \Delta \wedge q \xrightarrow{h''} p'' \text{ wrt } \Delta.$$

Since $D_{init}^{A, F_S} = \text{ldr}^\Delta(F \times (F_S \setminus F))$ by Lemma 10, there exist a nested word w and a pair $(q', q'') \in Q^2$ so that either (q', q'') or (q'', q') belongs to $F \times (F_S \setminus F)$ and

$$p' \xrightarrow{\text{hdg}(w)} q' \text{ wrt } \Delta \wedge p'' \xrightarrow{\text{hdg}(w)} q'' \text{ wrt } \Delta.$$

Hence, $nw(h) \cdot nw(h') \cdot w$ in L and $nw(h) \cdot nw(h'') \cdot w \in S \setminus L$ or vice versa. This shows that $nw(h)$ is relevant for L and S . $\square$

7.2.3. Updating the Difference Relation

For any difference relation $D \subseteq \mathcal{D}^\Delta$ and subset of states $Q \subseteq \mathcal{Q}$, we define a binary relation $down_Q^\Delta(D) \subseteq \mathcal{Q} \times \mathcal{Q}$, such that for all states $p_1, p_2 \in \mathcal{Q}$:

$$(p_1, p_2) \in down_Q^\Delta(D) \text{ iff } \exists q_1, q_2 \in Q. (q_1 @^\Delta p_1, q_2 @^\Delta p_2) \in D.$$

For any subset of states $Q \subseteq \mathcal{Q}$ and difference relation $D \in \mathcal{D}^\Delta$, let $D_Q \in \mathcal{D}^\Delta$ be the least difference relation that contains $down_Q^\Delta(D)$:

$$D_Q = ldr^\Delta(down_Q^\Delta(D)).$$

Lemma 11. The difference relation D_Q can be computed in time $O(|A|^2)$ from Q, Δ , and D .

Proof. The binary relation $down_Q^\Delta(D)$ can be computed in time $O(|\Delta|^2)$ from Q, D , and Δ . The least difference relation $ldr^\Delta(down_Q^\Delta(D))$ can be computed by ground datalog in time $O(|A|^2)$ from $down_Q^\Delta(D)$ using Proposition 4. $\square$

Example 11. Reconsider the dSHA for the XPath filter `[self::list/child::item]` in Figure 4 with the set of schema-final states $F_S = \{4, 5\}$. Since $F = \{4\}$, we have $F_S \setminus F = \{5\}$. The initial difference relation is $D_0 = D_{init}^{A, F_S}$, which is the symmetric closure of $(\{0\} \times \{1, 4, 5\}) \cup \{(1, 4), (4, 5)\}$. The subhedge irrelevant states are $Prj^\Delta(D_0) = \{1, 2, 3, 4, 5\}$ since only state 0 can access two states in the difference relation D_0 , the final state in $F = \{4\}$, and a nonfinal state that is schema-final in $F_S \setminus F = \{5\}$ of some hedges. The difference relation $down_{\{0\}}^\Delta(D_{init}^{A, F_S})$ is the symmetric closure of $\{1, 2\} \times \{3\}$, which is equal to the difference relation $D_1 = D_{0\{0\}} = ldr^\Delta(down_{\{0\}}^\Delta(D_0))$. Hence, the projection states are $Prj^\Delta(D_1) = \{2, 3, 4, 5\}$ since from states 0 and 1 one can still reach both 1 and 3 while $(1, 3) \in D_1$. The difference relation $D_2 = down_{\{1\}}^\Delta(D_1)$ is the symmetric closure of $\{(1, 2), (2, 3)\}$. Hence, the projection states are $Prj^\Delta(D_2) = \{1, 2, 3, 4, 5\}$. From state 1, one can only reach states 1 and 3, which are not different for D_2 .

Projection states for the initial difference relation contain all states that are safe for selection or safe for rejection with respect to the schema:

Lemma 12. All states in $safe^\Delta(F)$ and $safe^\Delta(F_S \setminus F)$ are subhedge irrelevant for D_{init}^{A, F_S} .

We omit the proof since the result is not used later on.

Given a dSHA $A = (\Sigma, \mathcal{Q}, \Delta, I, F)$ and a set $F \subseteq F_S \subseteq \mathcal{Q}$, we now construct the congruence projection $A^{cgr(S)}$ as the following dSHA[†]:

$$A^{cgr(S)} = (\Sigma, \mathcal{Q}^{cgr}, \Delta^{cgr}, I^{cgr(S)}, F^{cgr(S)}),$$

where the set of states, initial states, and final states are as follows:

$$\begin{aligned} \mathcal{Q}^{cgr} &\subseteq 2^{\mathcal{Q}} \times \mathcal{D}^\Delta \\ I^{cgr(S)} &= \{(I, D_{init}^{A, F_S})\} \\ F^{cgr(S)} &= \left\{ (Q, D_{init}^{A, F_S}) \mid \left(\begin{array}{l} Q \notin dPrj^\Delta(D_{init}^{A, F_S}) \Rightarrow Q \cap F \neq \emptyset \quad \wedge \\ Q \in dPrj^\Delta(D_{init}^{A, F_S}) \Rightarrow acc^\Delta(Q) \cap F \neq \emptyset \end{array} \right) \right\} \end{aligned}$$

The transition rules in Δ^{cgr} are provided by the inference rules in Figure 15. To illustrate construction and why determinization is needed, we reconsider Example 3, where schema restrictions have consequences that at first sight may be counter-intuitive.

$$\begin{array}{c}
\frac{a \in \Sigma \quad Q \in dPrj^\Delta(D)}{(Q, D) \xrightarrow{a} (Q, D) \text{ in } \Delta^{cgr}} \quad \frac{Q \in dPrj^\Delta(D)}{(Q, D) \xrightarrow{\langle \rangle} (Q, D) \text{ in } \Delta^{cgr}} \\
\\
\frac{Q @ P \rightarrow Q' \text{ in } \Delta^{det} \quad Q \in dPrj^\Delta(D)}{(Q, D) @ (Q, D) \rightarrow (Q, D) \text{ in } \Delta^{cgr}} \\
\\
\frac{Q \xrightarrow{a} Q' \text{ in } \Delta^{det} \quad Q \notin dPrj^\Delta(D)}{(Q, D) \xrightarrow{a} (Q', D) \text{ in } \Delta^{cgr}} \quad \frac{Q \notin dPrj^\Delta(D)}{(Q, D) \xrightarrow{\langle \rangle} (\langle \rangle^\Delta, D_Q) \text{ in } \Delta^{cgr}} \\
\\
\frac{Q @ acc^\Delta(P) \rightarrow Q' \text{ in } \Delta^{det} \quad Q \notin dPrj^\Delta(D) \quad P \in dPrj^\Delta(D_Q)}{(Q, D) @ (P, D_Q) \rightarrow (Q', D) \text{ in } \Delta^{cgr}} \\
\\
\frac{Q @ P \rightarrow Q' \text{ in } \Delta^{det} \quad Q \notin dPrj^\Delta(D) \quad P \notin dPrj^\Delta(D_Q)}{(Q, D) @ (P, D_Q) \rightarrow (Q', D) \text{ in } \Delta^{cgr}}
\end{array}$$

Figure 15. The transitions rules Δ^{cgr} of the congruence projection $A^{cgr}(S)$.

Example 12 (Determinization during congruence projection is needed.). Reconsider Example 3 with signature $\Sigma = \{a\}$, pattern $L = \{\langle a \rangle\}$, and schema $S = L \cup \{\langle \rangle \cdot a\}$. This language can be defined by the dSHA A in Figure 16. The schema S can be defined by the same automaton but with the schema-final states $F_S = \{3, 5\}$ and $F = \{3\}$. The dSHA $A^{cgr}(S)$ produced by congruence projection is shown in Figure 17. The unique hedge $\langle a \rangle$ of the language $L = \mathcal{L}(A)$ is accepted in state $(\{3, 4\}, D0)$, where $D0 = \{(3, 5), (5, 3)\}$. The unique hedge $\langle \rangle \cdot a$ in $S \setminus L$ is rejected: after reading the tree $\langle \rangle$, the automaton goes to state $(\{3, 4\}, D0)$ where it goes to a projecting sink $(\{5\}, D0)$ when reading the subsequent letter a . We note that any hedge in (Σ^*) is accepted in state $(\{3, 4\}, D0)$ by $A^{cgr}(S)$, even though only the single hedge $\langle a \rangle$ belongs to L . This is sound given that the hedges in $(\langle \epsilon + a \cdot a \cdot a^* \rangle)$ do not belong to schema S anyway. We notice that $(\{3, 4\}, D0)$ cannot be a projection state since the run on hedge $\langle \rangle \cdot a$ must continue to the sink $(\{5\}, D0)$; therefore, state $(\{3, 4\}, D0)$ is subhedge relevant.

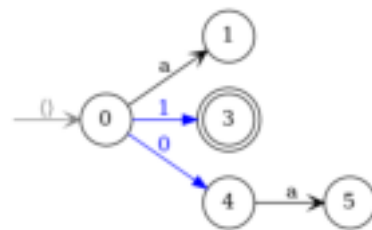


Figure 16. A dSHA A for $\mathcal{L}(A) = \{\langle a \rangle\}$ and schema-final states $F_S = \{3, 5\}$ for the schema $S = \mathcal{L}(A) \cup \{\langle \rangle \cdot a\}$.



Figure 17. The congruence projection $A^{cgr}(S)$ for the dSHA A and the schema-final states F_S in Figure 16.

Note that both individual states 3 and 4 are subhedge irrelevant for D_0 while the subset with both states $\{3, 4\}$ is subhedge relevant for D_0 since $(3, 5) \in \text{acc}^\Delta(\{3, 4\})^2 \cap D_0$. This shows that the deterministic construction is indeed needed to decide subhedge irrelevance for cases such as in this example. Also, notice that state 0 is subhedge irrelevant for $D_1 = \emptyset$. This is fine since the acceptance of hedges of the form $\langle h \rangle \cdot h'$ by $A^{\text{cgr}(S)}$ does indeed not depend on h . In contrast, it depends on h' , so that the subset of states $\{3, 4\}$ cannot be subhedge irrelevant for D_0 .

Finally, let us discuss how the transition rules of $d\text{SHA}^\downarrow A^{\text{cgr}(S)}$ are inferred. The most interesting transition rule is $(\{0\}, D_0) @ (\{0\}, D_1) \rightarrow (\{3, 4\}, D_0)$ in Δ^{cgr} . It is produced by the following inference rule, where $Q = P = \{0\}$, $\text{acc}^\Delta(P) = \{0, 1, 3, 4, 5\}$, $Q' = \{3, 4\}$, $D = D_0$, and $D_Q = D_1$:

$$\frac{Q @ \text{acc}^\Delta(P) \rightarrow Q' \text{ in } \Delta^{\text{det}} \quad Q \notin d\text{Prj}^\Delta(D) \quad P \in d\text{Prj}^\Delta(D_Q)}{(Q, D) @ (P, D_Q) \rightarrow (Q', D) \text{ in } \Delta^{\text{cgr}}}$$

This rule states that if P is subhedge irrelevant for D_Q , then all states accessible from P must be tried out since all of them could be reached by some different subhedge that got projected away. As a result, one cannot know due to subhedge projection into which state of Q' one should go. In order to stay deterministic, we thus go into all possible states, i.e., into the subset Q' . In the example, these are all the states that can be reached when reading some hedge in the pattern $\{\langle h \rangle \mid h \in \mathcal{H}_\Sigma\}$, given that the subhedges $h \in \Sigma$ are not inspected with subhedge projection.

Example 13 (XPath filter `[self::list/child::item]` with schema `[[doc]]`). The congruence projection of the example $d\text{SHA}$ in Figure 4 is given in Figure 18. This $d\text{SHA}$ is similar to the safe-no-change projection in Figure 7, except that the useless state 2' is removed and the four projection states are now looping in themselves rather than going to a shared looping state Π . It should also be noticed that only singleton state sets are used there, so no determinization is needed. As we will see, this is typical for experiments on regular XPath queries where larger subsets rarely occur.

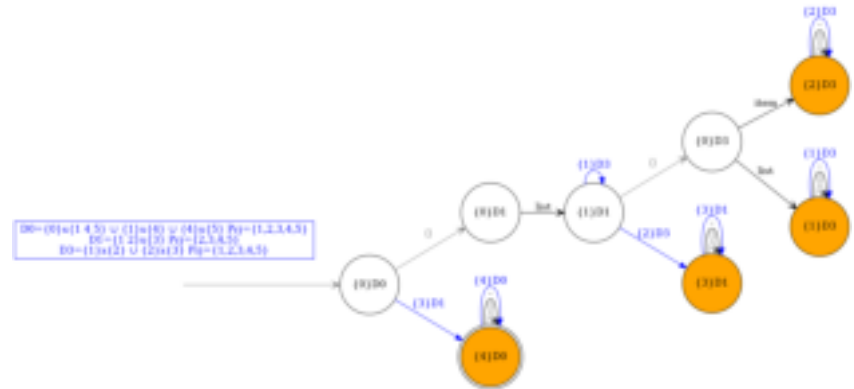


Figure 18. The congruence projection $A^{\text{cgr}(\llbracket \text{doc} \rrbracket)}$ of the $d\text{SHA}$ A in Figure 4 for the XPath filter `[self::list/child::item]` with schema `[[doc]]`. Subhedge irrelevant states are colored in orange. The underscore stands for any label, either `list` or `item`.

Example 14 (Counter-example for completeness of the safe-no-change projection.). Consider the counter example for the completeness of the safe-no-change projection, i.e., the $d\text{SHA}$ in Figure 12 for the XPath query `[child::item]` with schema final states $F_S = \{0, 5, 6\}$. Its congruence projection is shown in Figure 14. We note that the prefix `<item.item>` leads to the state $\{2\}D_3$, which is a subhedge projection state since 2 is subhedge irrelevant for D_3 . In particular, note that $\text{acc}^\Delta(\{2\}) = \{2, 4\}$ so no two states accessible from 2 are different in D_3 . This means that state 2 may still be changed to 4 but then it does not become different with respect to D_3 . This resolves the incompleteness issue with the safe-no-change projection on this example.

The first property for states (Q, D) assigned by the congruence projection is that Q is compatible with D . This means that no two states in Q are different with respect to D . Intuitively, each state of Q is as good as each other except for leading out of the schema. So, if (Q, D) is assigned to a prefix, and a suffix leads from some state in Q to F , then it cannot lead to $F_S \setminus F$ from some other state in Q .

Lemma 13. *If the partial run of $A^{cgr(S)}$ assigns a state (Q, D) to a prefix, then Q is compatible with D .*

We omit the proof since this instructive result will not be used directly later on. Still, compatibility will play an important role in the soundness proof.

7.3. Soundness

We next adapt the soundness result and proof from the safe-no-change projection to congruence projection.

Theorem 2 (Soundness of congruence projection.). *For any dSHA $A = (\Sigma, \mathcal{Q}, \Delta, I, F)$ and set $F \subseteq F_S \subseteq \mathcal{Q}$, the congruence projection of A with respect to F_S preserves the language of A within schema $S = \mathcal{L}(A[F/F_S])$, i.e.,*

$$\mathcal{L}(A^{cgr(S)}) \cap S = \mathcal{L}(A)$$

We note that S is a schema for A since $F \subseteq F_S$. Furthermore, A is schema-complete for S since A is deterministic and $S = \mathcal{L}(A[F/F_S])$.

Proof. The proof of the inclusion $\mathcal{L}(A) \subseteq \mathcal{L}(A^{cgr(S)}) \cap S$ will be based on the following two claims:

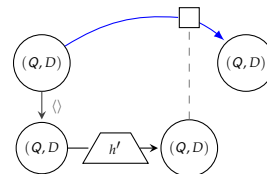
Claim 2.1a. For all $h \in \mathcal{H}_\Sigma$, $D \in \mathcal{D}^\Delta$, and $Q \in dPrj^\Delta(D)$, $(Q, D) \xrightarrow{h} (Q, D)$ wrt Δ^{cgr} .

Proof. We prove Claim 2.1a by induction on the structure of h . Suppose that $Q \in dPrj^\Delta(D)$.

Case $h = \langle h' \rangle$. The induction hypothesis applied to h' yields $(Q, D) \xrightarrow{h'} (Q, D)$ wrt Δ^{cgr} . We can thus apply the following two inference rules:

$$\frac{Q \in dPrj^\Delta(D)}{(Q, D) \xrightarrow{\emptyset} (Q, D) \text{ in } \Delta^{cgr}} \quad \frac{Q \in dPrj^\Delta(D)}{(Q, D) @ (Q, D) \rightarrow (Q, D) \text{ in } \Delta^{cgr}}$$

in order to close the following diagram with respect to Δ^{cgr} :



This proves $(Q, D) \xrightarrow{h} (Q, D)$ wrt Δ^{cgr} , as required by the claim.

Case $h = a$. Since $q \in dPrj^\Delta(D)$, we can apply the inference rule:

$$\frac{a \in \Sigma \quad Q \in dPrj^\Delta(D)}{(Q, D) \xrightarrow{a} (Q, D) \text{ in } \Delta^{cgr}}$$

This proves this case of the claim.

Case $h = \varepsilon$. We trivially have $(Q, D) \xrightarrow{\varepsilon} (Q, D) \text{ wrt } \Delta^{cgr}$.

Case $h = h' \cdot h''$. By the induction hypothesis applied to h' and h'' , we obtain the following:

$$(Q, D) \xrightarrow{h'} (Q, D) \text{ and } (Q, D) \xrightarrow{h''} (Q, D) \text{ wrt } \Delta^{cgr}. \text{ Hence, } (Q, D) \xrightarrow{h' \cdot h''} (Q, D) \text{ wrt } \Delta^{cgr}.$$

This ends the proof of Claim 2.1a. The next claim is the key of the soundness proof. $\square$

For any difference relation D , we define a binary relation $dep^a(D) \subseteq 2^Q \times 2^Q$, such that, for any two subsets of states $Q', Q'' \subseteq Q$,

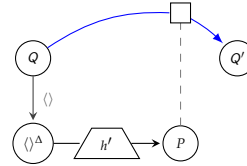
$$(Q', Q'') \in dep^a(D) \Leftrightarrow \begin{cases} Q' \notin dPrj^\Delta(D) \Rightarrow Q' \subseteq Q'' & \wedge \\ Q' \in dPrj^\Delta(D) \Rightarrow (Q' \subseteq acc^\Delta(Q'') \wedge Q'' \in dPrj^\Delta(D)) & \end{cases} \quad \begin{matrix} (1^a) \\ (2^a) \end{matrix}$$

Furthermore, note that $Q'' \in dPrj^\Delta(D) \wedge Q' \subseteq acc^\Delta(Q'')$ implies $Q' \in dPrj^\Delta(D)$, and thus, $(Q', Q'') \in dep^a(D)$.

Claim 2.2a. Let $h \in \mathcal{H}_\Sigma$ a hedge, $Q, Q' \subseteq Q$ subsets of states, and $D \in \mathcal{D}^\Delta$ a difference relation. If $Q \xrightarrow{h} Q' \text{ wrt } \Delta^{det}$, then there exists $Q'' \subseteq Q$, such that $(Q, D) \xrightarrow{h} (Q'', D) \text{ wrt } \Delta^{cgr}$ and $(Q', Q'') \in dep^a(D)$.

Proof. If $Q \in dPrj^\Delta(D)$, then $(Q, D) \xrightarrow{h} (Q, D) \text{ wrt } \Delta^{cgr}$ by Claim 2.1a. Let $Q'' = Q$. We then have $Q' \subseteq acc^\Delta(Q'')$ and $Q'' \in dPrj^\Delta(D)$, and thus, $Q' \in dPrj^\Delta(D)$ so that (1^a) and (2^a) of $(Q', Q'') \in dep^a(D)$. So, it is sufficient to prove the claim under the assumption that $Q \notin dPrj^\Delta(D)$. The proof is by induction on the structure of h .

Case $h = \langle h' \rangle$. The assumption $Q \xrightarrow{h} Q' \text{ wrt } \Delta^{det}$ shows that there exists a subset of states $P \subseteq Q$ closing the following diagram:



In particular, we have $Q@P \rightarrow Q' \text{ in } \Delta^{det}$. Let $D' = D_Q$. Since $Q \notin dPrj^\Delta(D)$, we can infer the following:

$$\frac{Q \notin dPrj^\Delta(D)}{(Q, D) \xrightarrow{\langle \rangle} (\langle \rangle^\Delta, D') \text{ in } \Delta^{cgr}}$$

We have to distinguish two cases depending on whether $\langle \rangle^\Delta$ belongs to $dPrj^\Delta(D')$ or not.

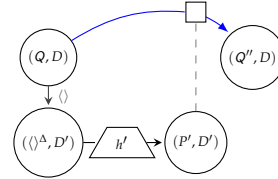
Subcase $\langle \rangle^\Delta \notin dPrj^\Delta(D')$. The induction hypothesis applied to h' yields the existence of $P' \subseteq Q$, such that $(P, P') \in dep^a(D')$ and

$$(\langle \rangle^\Delta, D') \xrightarrow{h'} (P', D') \text{ wrt } \Delta^{cgr}.$$

Subsubcase $P \notin dPrj^\Delta(D')$. Since $(P, P') \in dep^a(D')$, it follows that $P \subseteq P'$. Hence, $P' \notin dPrj^\Delta(D')$. Let $Q'' \subseteq Q$ be such that $Q@P' \rightarrow Q'' \text{ in } \Delta^{det}$. We can then apply the following inference rule:

$$\frac{Q@P' \rightarrow Q'' \text{ in } \Delta^{det} \quad Q \notin dPrj^\Delta(D) \quad P' \notin dPrj^\Delta(D')}{(Q, D)@ (P', D') \rightarrow (Q'', D) \text{ in } \Delta^{cgr}}$$

Hence, we can close the diagram as follows:

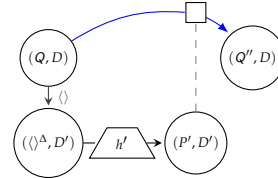


This shows that $(Q, D) \xrightarrow{h} (Q'', D)$ wrt Δ^{cgr} . Since $P \subseteq P'$, $Q@P \rightarrow Q'$, and $Q@P' \rightarrow Q''$ in Δ^{det} , it follows that $Q' \subseteq Q''$, and thus, $(Q', Q'') \in dep^a(D)$. This shows the claim in this case.

Subsubcase $P \in dPrj^\Delta(D')$. Since $(P, P') \in dep^a(D')$, it follows that $P \subseteq acc^\Delta(P')$ and $P' \in dPrj^\Delta(D')$. Hence, we can apply the following inference rule for some $Q'' \subseteq \mathcal{H}_\Sigma$:

$$\frac{Q@acc^\Delta(P') \rightarrow Q'' \text{ in } \Delta^{det} \quad Q \notin dPrj^\Delta(D) \quad P' \in dPrj^\Delta(D')}{(Q, D)@(P', D') \rightarrow (Q'', D) \text{ in } \Delta^{cgr}}$$

Hence, we can close the diagram as follows:

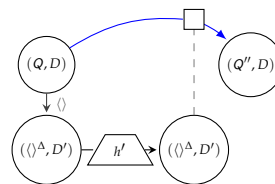


This shows that $(Q, D) \xrightarrow{h} (Q'', D)$ wrt Δ^{cgr} . Since $P \subseteq acc^\Delta(P')$, it follows from $Q@P \rightarrow Q'$ and $Q@acc^\Delta(P') \rightarrow Q''$ in Δ^{det} that $Q' \subseteq Q''$. Thus, $(Q', Q'') \in dep^a(D)$, so the claim holds in this case too.

Subcase $\langle \rangle^\Delta \in Prj^\Delta(D')$. Claim 2.1a shows that $(\langle \rangle^\Delta, D') \xrightarrow{h'} (\langle \rangle^\Delta, D')$ wrt Δ^{cgr} . Let Q'' be such that $Q@acc^\Delta(\langle \rangle^\Delta) \rightarrow Q''$ in Δ^{det} . We can apply the following inference rule:

$$\frac{Q@acc^\Delta(\langle \rangle^\Delta) \rightarrow Q'' \text{ in } \Delta^{det} \quad Q \notin dPrj^\Delta(D) \quad \langle \rangle^\Delta \in Prj^\Delta(D')}{(Q, D)@(\langle \rangle^\Delta, D') \rightarrow (Q'', D) \text{ in } \Delta^{cgr}}$$

We can thus close the diagram below as follows with respect to Δ^{cgr} :



This shows that $(Q, D) \xrightarrow{h} (Q'', D)$ wrt Δ^{cgr} . Since $P \subseteq acc^\Delta(\langle \rangle^\Delta)$, it follows from $Q@P \rightarrow Q'$ and $Q@acc^\Delta(\langle \rangle^\Delta) \rightarrow Q''$ in Δ^{det} that $Q' \subseteq Q''$, and thus, $(Q', Q'') \in dep^a(D)$.

Case $h = a$. Since $Q \notin dPrj^\Delta(D)$, we can apply the inference rule:

$$\frac{Q \xrightarrow{a} Q' \text{ in } \Delta^{det} \quad Q \notin dPrj^\Delta(D)}{(Q, D) \xrightarrow{a} (Q', D) \text{ in } \Delta^{cgr}}$$

Hence, $(Q, D) \xrightarrow{h} (Q', D)$ wrt Δ^{cgr} . With $Q'' = Q'$, the claim follows since $(Q', Q'') \in dep^a(D)$.

Case $h = \varepsilon$. In this case, we have $Q = Q''$ and $(Q, D) \xrightarrow{\varepsilon} (Q, D)$ wrt Δ^{cgr} , so the claim holds.

Case $h = h_1 \cdot h_2$. Since $Q \xrightarrow{h} Q'$ wrt Δ^{det} , there exists $Q_1 \subseteq Q$, such that $Q \xrightarrow{h_1} Q_1$ and $Q_1 \xrightarrow{h_2} Q'$ wrt Δ^{det} . By the induction hypothesis applied to h_1 there exists $Q'_1 \subseteq Q$, such that $(Q, D) \xrightarrow{h_1} (Q'_1, D)$ wrt Δ^{cgr} and $(Q_1, Q'_1) \in dep^a(D)$.

Subcase $Q_1 \in dPrj^\Delta(D)$. Since $(Q_1, Q'_1) \in dep^a(D)$, it follows that $Q_1 \subseteq acc^\Delta(Q'_1)$ and $Q'_1 \in dPrj^\Delta(D)$. Furthermore, Claim 2.1a shows that $(Q'_1, D) \xrightarrow{h_2} (Q'_1, D)$, and hence, $(Q, D) \xrightarrow{h} (Q'_1, D)$ wrt Δ^{cgr} . Since $Q' \subseteq acc^\Delta(Q_1)$ and $Q_1 \subseteq acc^\Delta(Q'_1)$, it follows that $Q' \subseteq acc^\Delta(Q'_1)$. Hence, $(Q'_1, Q') \in dep^a(D)$ and $(Q, D) \xrightarrow{h} (Q'_1, D)$.

Subcase $Q_1 \notin dPrj^\Delta(D)$. In this case, we can apply the induction hypothesis to $Q_1 \xrightarrow{h_2} Q'$ wrt Δ^{det} , showing that there exists $Q'' \subseteq Q$, such that $(Q'_1, D) \xrightarrow{h_2} (Q'', D)$ wrt Δ^{cgr} and $(Q', Q'') \in dep^a(D)$. Hence, $(Q, D) \xrightarrow{h} (Q'', D)$ wrt Δ^{cgr} and $(Q', Q'') \in dep^a(D)$.

This ends the proof of Claim 2.2a. $\square$

Proof of Inclusion $\mathcal{L}(A) \subseteq \mathcal{L}(A^{cgr(S)}) \cap \mathbf{S}$. Since $\mathbf{S}$ is a schema for A , we have $\mathcal{L}(A) \subseteq \mathbf{S}$, so that it is sufficient to show $\mathcal{L}(A) \subseteq \mathcal{L}(A^{cgr(S)})$. Let $h \in \mathcal{L}(A)$. Then, there exist $q_0 \in I$ and $q \in F$, such that $q_0 \xrightarrow{h} q$ wrt Δ . Let $Q = \{q\}$. Since A is deterministic, it follows that $I \xrightarrow{h} Q$ wrt Δ^{det} . Using Claim 2.2a, this implies the existence of a subset of states $Q' \subseteq Q$, such that $(Q, Q') \in dep^a(D)$ and $(I, D_{init}^{A, Fs}) \xrightarrow{h} (Q', D_{init}^{A, Fs})$ wrt Δ^{cgr} . Furthermore, $(I, D_{init}^{A, Fs}) \in I^{cgr(S)}$. In order to prove $h \in \mathcal{L}(A^{cgr(S)})$, it is thus sufficient to show that $(Q', D_{init}^{A, Fs}) \in F^{cgr(S)}$. We distinguish two cases:

Case $Q' \notin dPrj^\Delta(D_{init}^{A, Fs})$. Condition (1^a) of $(Q, Q') \in dep^a(D)$, shows that $Q \subseteq Q'$ so that $q \in Q'$. Since $q \in F$, it follows that $Q' \cap F \neq \emptyset$. Thus, $(Q', D_{init}^{A, Fs}) \in F^{cgr(S)}$, so that $h \in \mathcal{L}(A^{cgr(S)})$.

Case $Q' \in dPrj^\Delta(D_{init}^{A, Fs})$. Condition (2^a) of $(Q, Q') \in dep^a(D)$ yields $Q \subseteq acc^\Delta(Q')$. Hence, $q \in acc^\Delta(Q') \cap F$ so that $(Q', D_{init}^{A, Fs}) \in F^{cgr(S)}$, and thus, $h \in \mathcal{L}(A^{cgr(S)})$.

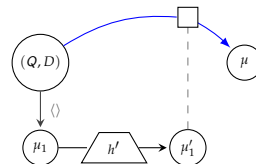
This ends the proof of the first inclusion.

We next want to show the inverse inclusion $\mathcal{L}(A^{cgr(S)}) \cap \mathbf{S} \subseteq \mathcal{L}(A)$. It will eventually follow from the next two claims.

Claim 2.1b. For any hedge $h \in \mathcal{H}_\Sigma$, difference relation $D \subseteq \mathcal{D}^\Delta$, projection state $Q \in dPrj^\Delta(D)$ and state $\mu \in \mathcal{Q}^{cgr}$: if $(Q, D) \xrightarrow{h} \mu$ wrt Δ^{cgr} , then $\mu = (Q, D)$.

Proof. By induction on the structure of $h \in \mathcal{H}_\Sigma$, suppose that $(Q, D) \xrightarrow{h} \mu$ wrt Δ^{cgr} :

Case $h = \langle h' \rangle$. There must exist states $\mu_1, \mu'_1 \in \mathcal{Q}^{cgr}$ closing the following diagram:



Since $Q \in dPrj^\Delta(D)$, the following rule must have been applied to infer $(Q, D) \xrightarrow{\langle h' \rangle} \mu_1$ wrt Δ^{cgr} :

$$\frac{Q \in dPrj^\Delta(D)}{(Q, D) \xrightarrow{\langle h' \rangle} (Q, D) \text{ in } \Delta^{cgr}}$$

Therefore, $\mu_1 = (Q, D)$. The induction hypothesis applied to $(Q, D) \xrightarrow{h'} \mu'_1$ wrt Δ^{cgr} shows that $\mu'_1 = (Q, D)$, too. So, μ must have been inferred by applying the rule:

$$\frac{Q \in dPrj^\Delta(D)}{(Q, D) @ (Q, D) \rightarrow (Q, D) \text{ in } \Delta^{cgr}}$$

Hence, $\mu = (Q, D)$, as required.

Case $h = a$. The following rule must be applied:

$$\frac{a \in \Sigma \quad Q \in dPrj^\Delta(D)}{(Q, D) \xrightarrow{a} (Q, D) \text{ in } \Delta^{cgr}}$$

Hence, $\mu = (Q, D)$.

Case $h = \varepsilon$. Obvious.

Case $h = h_1 \cdot h_2$. There must exist some μ_1 , such that $(Q, D) \xrightarrow{h_1} \mu_1 \xrightarrow{h_2} \mu$ wrt Δ^{cgr} . Using the induction hypothesis applied to h_1 , we have $\mu_1 = (Q, D)$. We can thus apply the induction hypothesis to h_2 to obtain $\mu_2 = (Q, D)$.

This ends the proof of Claim 2.1b. $\square$

We next need an inverse of Claim 2.2a. For relating runs A^{cgr} back to runs of A , we define for any difference relation D another binary relation $dep^b(D) \subseteq 2^Q \times 2^Q$, such that for any two subsets of states $Q', Q'' \subseteq Q$,

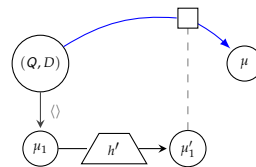
$$(Q', Q'') \in dep^b(D) \Leftrightarrow \begin{cases} (Q' \times Q'') \cap D = \emptyset & \wedge & (0^b) \\ Q' \notin dPrj^\Delta(D) \Rightarrow Q'' \subseteq Q' & \wedge & (1^b) \\ Q' \in dPrj^\Delta(D) \Rightarrow Q'' \subseteq acc^\Delta(Q') & (2^b) \end{cases}$$

Claim 2.2b. Let $Q \in \mathcal{Q}$ be a subset of states that is compatible with a difference relation $D \in \mathcal{D}^\Delta$. For any hedge $h \in \mathcal{H}_\Sigma$ and state $\mu \in \mathcal{Q}^{cgr}$ with $(Q, D) \xrightarrow{h} \mu$ wrt Δ^{cgr} , there exist a pair of subsets of states $(Q', Q'') \in dep^b(D)$, such that $\mu = (Q', D)$ and $Q \xrightarrow{h} Q''$ wrt Δ^{det} .

Proof. If $Q \in dPrj^\Delta(D)$, then Claim 2.1b shows that $\mu = (Q, D)$. Let $Q' = Q$ and Q'' be the unique subset of states, such that $Q \xrightarrow{h} Q''$ wrt Δ^{det} . We have to show that $(Q', Q'') \in dep^b(D)$. Since $Q \xrightarrow{h} Q''$, we have $Q'' \subseteq acc^\Delta(Q')$, so condition (2^b) holds. Condition (1^b) holds trivially since $Q \in dPrj^\Delta(D)$. For condition (0^b) , note that $Q \in dPrj^\Delta(D)$ implies $acc^\Delta(Q)^2 \cap D = \emptyset$. Furthermore, $Q' \times Q'' \subseteq acc^\Delta(Q)^2$, so that $(Q' \times Q'') \cap D = \emptyset$ as required.

We can thus assume that $Q \notin dPrj^\Delta(D)$. The proof is by induction on the structure of $h \in \mathcal{H}_\Sigma$. We distinguish all the possible forms of hedges $h \in \mathcal{H}_\Sigma$:

Case $h = \langle h' \rangle$. By definition of $(Q, P) \xrightarrow{h} \mu$ wrt Δ^{cgr} , there must exist $\mu_1, \mu'_1 \in \mathcal{Q}^{cgr}$, such that the following diagram can be closed:



Since $Q \notin dPrj^\Delta(D)$, the following inference rule was applied to infer $(Q, D) \xrightarrow{\langle \rangle} \mu_1$ wrt Δ^{cgr} , where $D' = D_Q$:

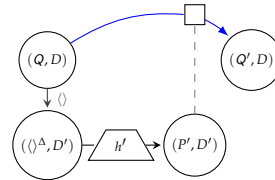
$$\frac{Q \notin dPrj^\Delta(D)}{(Q, D) \xrightarrow{\langle \rangle} (\langle \rangle^\Delta, D') \text{ in } \Delta^{cgr}}$$

Hence, $\mu_1 = (\langle \rangle^\Delta, D')$. If $\langle \rangle^\Delta \notin dPrj^\Delta(D)$, then the induction hypothesis applied to $(\langle \rangle^\Delta, D') \xrightarrow{h'} \mu'_1$ wrt Δ^{cgr} shows that there exists $(P', P'') \in dep^b(D')$, such that $\mu'_1 = (P', D')$ and $\langle \rangle^\Delta \xrightarrow{h'} P''$ wrt Δ^{det} . Otherwise, the same can be concluded as we argued at the beginning.

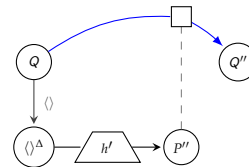
Subcase $P' \in dPrj^\Delta(D')$. The transition rule $(Q, D)@_{\mu_1} \rightarrow \mu$ must be inferred as follows for $Q' \subseteq Q$:

$$\frac{Q@acc^\Delta(P') \rightarrow Q' \text{ in } \Delta^{det} \quad Q \notin dPrj^\Delta(D) \quad P' \in dPrj^\Delta(D)}{(Q, D)@_{(P', D')} \rightarrow (Q', D) \text{ in } \Delta^{cgr}}$$

This shows that $\mu = (Q', D)$. So, we have the following diagram:



Let Q'' be the unique subset of states, such that $Q@P'' \rightarrow Q''$ wrt Δ^{det} . We can then close the following diagram:

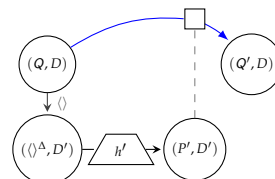


From $(P', P'') \in dep^b(D')$, it follows $(P' \times P'') \cap D' = \emptyset$, and thus, $(Q' \times Q'') \cap D = \emptyset$, i.e., condition (0^b) of $(Q', Q'') \in dep^b(D)$. Since $P' \in dPrj^\Delta(D')$, condition (2^b) of $(P', P'') \in dep^b(D')$ yields $P'' \subseteq acc^\Delta(P')$. Furthermore, $Q@P'' \rightarrow Q''$ and $Q@acc^\Delta(P') \rightarrow Q'$ in Δ^{det} , which yields $Q'' \subseteq Q'$ so that conditions (1^b) and (2^b) of $(Q', Q'') \in dep^b(D)$ follow. Hence, $(Q', Q'') \in dep^b(D)$. In summary, we show that $\mu = (Q', D)$, $Q \xrightarrow{\langle h' \rangle} Q''$ wrt Δ^{det} , and $(Q', Q'') \in dep^b(D)$, as required by the claim.

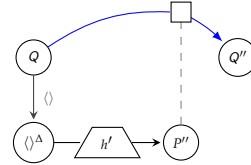
Subcase $P' \notin dPrj^\Delta(D')$. The transition rule $(Q, D)@_{\mu_1} \rightarrow \mu$ must thus be inferred as follows for $Q' \subseteq Q$:

$$\frac{Q@P' \rightarrow Q' \text{ in } \Delta^{det} \quad Q \notin dPrj^\Delta(D) \quad P' \notin dPrj^\Delta(D)}{(Q, D)@_{(P', D)} \rightarrow (Q', D) \text{ in } \Delta^{cgr}}$$

This shows that $\mu = (Q', D)$, and we can close the following diagram:



Condition (0^b) of $(P', P'') \in \text{dep}^b(D')$ yields $(P' \times P'') \cap D' = \emptyset$. Let Q'' be the unique subset of states, such that $Q@P'' \rightarrow Q''$ wrt. Δ^{det} . Since $D' \subseteq \text{down}_Q^\Delta(D)$ and $(P' \times P'') \cap D' = \emptyset$, it follows from $Q@P' \rightarrow Q'$ and $Q@P'' \rightarrow Q''$ wrt. Δ^{det} that $(Q' \cap Q'') \cap D = \emptyset$. That is, condition (0^b) of $(Q', Q'') \in \text{dep}^b(D)$. Since $P' \notin d\text{Prj}^\Delta(D')$, condition (1^b) of $(P', P'') \in \text{dep}^b(D')$ is $P'' \subseteq P'$. From $Q@P'' \rightarrow Q''$ and $Q@P' \rightarrow Q'$, it thus follows that $Q'' \subseteq Q'$. Hence, conditions (1^b) and (2^b) of $(Q', Q'') \in \text{dep}^b(D)$ are valid, so that $(Q', Q'') \in \text{dep}^b(D)$ holds. Furthermore,



This shows that $Q \xrightarrow{\langle h' \rangle} Q''$ wrt Δ . The other two requirements of the claim, $\mu = (Q', D)$ and $(Q', Q'') \in \text{dep}^b(D)$, were shown earlier.

Case $h = a$. Since $Q \notin d\text{Prj}^\Delta(D)$, the following inference rule must be used:

$$\frac{Q \xrightarrow{a} Q' \text{ in } \Delta \quad Q \notin d\text{Prj}^\Delta(D)}{(Q, D) \xrightarrow{a} (Q', D) \text{ in } \Delta^{\text{cgr}}}$$

So $\mu = (Q', D)$, $Q \xrightarrow{a} Q'$ wrt Δ^{det} . Furthermore, since Q is compatible with D , and D is a difference relation, Q' is compatible with D too. Hence, $(Q' \times Q') \cap D = \emptyset$, showing condition (0^b) of $(Q', Q') \in \text{dep}^b(D)$. Trivially, $Q' \subseteq Q' \subseteq \text{acc}^\Delta(Q')$ so conditions (1^b) and (2^b) of $(Q', Q') \in \text{dep}^b(D)$ hold, too. Hence, $(Q', Q') \in \text{dep}^b(D)$.

Case $h = \varepsilon$. Obvious.

Case $h = h_1 \cdot h_2$. Since $(Q, D) \xrightarrow{h} \mu$ wrt. Δ^{cgr} , there exists some $\mu_1 \in \mathcal{Q}^{\text{cgr}}$, such that $(Q, D) \xrightarrow{h_1} \mu_1$ and $\mu_1 \xrightarrow{h_2} \mu$ wrt Δ^{cgr} . We can apply the induction hypothesis to h_1 .

Hence, there exist subsets of states $Q_1, Q'_1 \subseteq \mathcal{Q}$, such that $\mu_1 = (Q_1, D)$, $Q \xrightarrow{h_1} Q'_1$ wrt Δ , and $(Q_1, Q'_1) \in \text{dep}^b(D)$. We distinguish two cases:

Subcase $Q_1 \in d\text{Prj}^\Delta(D)$. Since $\mu_1 = (Q_1, D) \xrightarrow{h_2} \mu$ wrt Δ^{cgr} , Claim 2.1b shows that $\mu = (Q_1, D)$ so that $(Q_1, D) \xrightarrow{h_2} (Q_1, D)$ wrt. Δ^{cgr} . Condition (2^b) of $(Q_1, Q'_1) \in \text{dep}^b(D)$ and $Q_1 \in d\text{Prj}^\Delta(D)$ imply that $Q'_1 \in \text{acc}^\Delta(Q_1)$. Let Q_2 be the unique subset of states, such that $Q'_1 \xrightarrow{h_2} Q_2$ wrt. Δ^{det} . Then, $Q_2 \subseteq \text{acc}^\Delta(Q'_1)$ so that $Q_2 \subseteq \text{acc}^\Delta(Q_1)$, and thus, condition (2^b) of $(Q_1, Q_2) \in \text{dep}^b(D)$ holds. Condition (1^b) of $(Q_1, Q_2) \in \text{dep}^b(D)$ is trivial since $Q_1 \in d\text{Prj}^\Delta(D)$. Since $Q_1 \in d\text{Prj}^\Delta(D)$ and $Q_1 \times Q_2 \in \text{acc}^\Delta(Q)^2$, it follows that $Q_1 \times Q_2 \cap D = \emptyset$ so condition (0^b) of $(Q_1, Q_2) \in \text{dep}^b(D)$ holds, too. Hence, $Q \xrightarrow{h} Q_2$ wrt Δ^{det} and $(Q_1, Q_2) \in \text{dep}^b(D)$.

Subcase $Q_1 \notin d\text{Prj}^\Delta(D)$. Since $Q_1 \xrightarrow{h_1} Q'_1$ and Q_1 is compatible with difference relation D , it follows that Q'_1 is compatible with D , too. We can thus apply the induction hypothesis to $(Q_1, D) \xrightarrow{h_2} \mu$, showing the existence of subsets of states $(Q_2, Q'_2) \in \text{dep}^b(D)$, such that $\mu = (Q_2, D)$ and $Q_1 \xrightarrow{h_2} Q'_2$ wrt Δ . So, we have $Q \xrightarrow{h} Q'_2$ wrt. Δ and $(Q_2, Q'_2) \in \text{dep}^b(D)$, as required.

This ends the proof of Claim 2.2b. $\square$

Proof of Inclusion $\mathcal{L}(A^{\text{cgr}}(\mathcal{S})) \cap \mathcal{S} \subseteq \mathcal{L}(A)$. Let $h \in \mathcal{L}(A^{\text{cgr}}(\mathcal{S})) \cap \mathcal{S}$. Then, there exists a final subset of states $Q \in F^{\text{cgr}}(\mathcal{S})$, such that $(I, D_{\text{init}}^{A, \text{Fs}}) \xrightarrow{h} (Q, D_{\text{init}}^{A, \text{Fs}})$ wrt Δ^{cgr} . Since I is

a singleton or empty, it is compatible with D_{init}^{A, F_S} . Claim 2.2b shows that there exists a subset of states $Q' \subseteq Q$, such that $I \xrightarrow{h} Q'$ wrt. Δ^{det} and $(Q, Q') \in dep^b(D_{init}^{A, F_S})$. Condition (0^b) of $(Q, Q') \in dep^b(D_{init}^{A, F_S})$ shows that $(Q, Q') \cap D_{init}^{A, F_S} = \emptyset$. Since $h \in S$, it follows that $Q' \cap F_S \neq \emptyset$. The determinism of A shows that Q' is a singleton. So, there exists a state $q' \in F_S$, such that $Q' = \{q'\}$.

Case $Q \notin dPrj^\Delta(D_{init}^{A, F_S})$. Condition (1^b) shows that $Q' \subseteq Q$ so that $q' \in Q$. Since $Q \in F_S^{cgr}$, we have $Q \cap F \neq \emptyset$. Since Q is compatible with D_{init}^{A, F_S} , $q' \in Q$, and $Q \cap F \neq \emptyset$, it follows that $q' \notin F_S \setminus F$. Since $q' \in F_S$, this implies $q' \in F$, and thus, $h \in \mathcal{L}(A)$.

Case $Q \in dPrj^\Delta(D_{init}^{A, F_S})$. Condition (2^b) shows that $Q' \subseteq acc^\Delta(Q)$ so that $q' \in acc^\Delta(Q)$. Since $Q \in F_S^{cgr}$, we have $acc^\Delta(Q) \cap F \neq \emptyset$. Let $q \in acc^\Delta(Q) \cap F$ be arbitrary. Since $Q \in dPrj^\Delta(D_{init}^{A, F_S})$ and $(q', q) \in acc^\Delta(Q)^2$, it follows that $(q, q') \notin D_{init}^{A, F_S}$. Furthermore, $q \in F$ so that $q' \notin F_S \setminus F$. In combination with $q' \in F_S$, this implies $q' \in F_S$ so $h \in \mathcal{L}(A)$.

This ends the proof of the inverse inclusion, and thus, of $\mathcal{L}(A) = \mathcal{L}(A^{cgr(S)}) \cap S$. $\square$

7.4. Completeness

We next show the completeness of congruence projection for subhedge projection according to Definition 10. Let $A = (\Sigma, Q, \Delta, I, F)$ be a complete dSHA and $F \subseteq F_S \subseteq Q$. Automaton A defines the regular pattern $L = \mathcal{L}(A)$ with the schema-final states in F_S the regular schema $S = \mathcal{L}(A[F/F_S])$. We first show that all subhedge irrelevant states of difference relations are subhedge projection states $A^{cgr(S)}$ according to Definition 7.

Lemma 14. *If $Q \in dPrj^\Delta(D)$, then (Q, D) is a subhedge projection state of Δ^{cgr} with witness (Q, D) .*

Proof. We assume that $q \in Prj^\Delta(D)$ and have to show that (Q, D) is a subhedge projection state of Δ^{cgr} . We have to show that all transition rules starting with (Q, D) are permitted for a subhedge projection state (Q, D) with (Q, D) . They are generated by the following rules, all of which are looping:

$$\frac{a \in \Sigma \quad Q \in dPrj^\Delta(D)}{(Q, D) \xrightarrow{a} (Q, D) \text{ in } \Delta^{cgr}} \quad \frac{Q \in dPrj^\Delta(D)}{(Q, D) @ (Q, D) \rightarrow (Q, D) \text{ in } \Delta^{cgr}} \quad \frac{Q \in dPrj^\Delta(D)}{(Q, D) \xrightarrow{\emptyset} (Q, D) \text{ in } \Delta^{cgr}}$$

$\square$

So, if a partial run of $A^{cgr(S)}$ on a prefix u assigns some state (P, D) with $P \in dPrj^\Delta(D)$, then (P, D) is a subhedge projection state by Lemma 14, and thus, subhedge irrelevant by Proposition 2.

Lemma 15. *Let $A' = A[I/\langle \rangle^\Delta]$, $L' = \mathcal{L}(A')$, and $S' = \mathcal{L}(A'[F/F_S])$. If $(\langle \rangle^\Delta, D) \xrightarrow{hdg(u)} (Q, D)$ wrt $A^{cgr(S)}$ for some nested word $u \in \mathcal{N}_\Sigma$ and either*

- $Q \notin dPrj^\Delta(D)$ and $q \in Q$, or
- $Q \in dPrj^\Delta(D)$ and $q \in acc^\Delta(Q)$,

then there exists a nested word $u' \in class_{S'}^{L'}(u)$ and $q_0 \in \langle \rangle^\Delta$, such that $q_0 \xrightarrow{hdg(u')} q$ wrt Δ .

Proof. By induction on the length of u , suppose that $(\langle \rangle^\Delta, D) \xrightarrow{hdg(u)} (Q, D)$ wrt $A^{cgr(S)}$. In the base case, we have $u = \varepsilon$. Hence, $Q = \langle \rangle^\Delta$.

Case $Q \notin dPrj^\Delta(D)$ and $q \in Q$. Then, $\langle \rangle^\Delta = \{q\}$. The unique run of A' on $u = \varepsilon$ starts and ends in the tree initial state $q \in \langle \rangle^\Delta$.

Case $Q \in dPrj^\Delta(D)$ and $q \in acc^\Delta(Q)$. Since $Q \in dPrj^\Delta(D)$, Lemma 14 shows that (Q, D) is a subhedge projection state, so that ε is subhedge irrelevant for L' and S' by Proposition 2. Hence, $class_{S'}^{L'}(\varepsilon) = \mathcal{N}_\Sigma$. Furthermore, since $q \in acc^\Delta(Q)$, there exists a hedge h and $q_0 \in I$, such that $q_0 \xrightarrow{h} q$ wrt Δ . Let $u' = nw(h)$. The run of A on u' ends in q and $u' \in class_{S'}^{L'}(u)$.

For the induction step, we distinguish the possible forms of the nested word u . So, there exist $\tilde{u}, v \in \mathcal{N}_\Sigma$ and $a \in \Sigma$, such that either of the following cases holds:

Case $u = \tilde{u} \cdot \langle \cdot v \cdot \rangle$. Let $(\tilde{Q}, D)$ be the state in which the run of $(A')^{cgr(S)}$ on $\tilde{u}$ ends.

Subcase $\tilde{Q} \notin dPrj^\Delta(D)$. Let $D' = D_{\tilde{Q}}$. Then, there exist $P \subseteq Q$, such that $(\langle \rangle^\Delta, D') \xrightarrow{v} (P, D')$ wrt $A^{cgr(S)}$. So, the run of $(A')^{cgr(S)}$ on v goes to state (P, D') .

Subsubcase $P \in dPrj^\Delta(D')$. By construction of $A^{cgr(S)}$, we then have $Q = \tilde{Q} @^\Delta acc^\Delta(P)$. Since $q \in Q$, there exist $\tilde{q} \in \tilde{Q}$ and $p \in acc^\Delta(P)$, such that $\tilde{q} @ p \rightarrow q$ in Δ . By induction hypothesis, there exists $v' \in class_{S'}^{L'}(v)$, such that the run of A' on v' ends in p . Hence, the run of A on $u' = \tilde{u} \cdot \langle \cdot v' \cdot \rangle$ ends in q , and furthermore, $u' \in class_S^L(u)$.

Subsubcase $P \notin dPrj^\Delta(D')$. By construction of $A^{cgr(S)}$, we then have $Q = \tilde{Q} @^\Delta P$. Since $q \in Q$, there exist $\tilde{q} \in \tilde{Q}$ and $p \in P$, such that $\tilde{q} @ p \rightarrow q$ in Δ . By induction hypothesis, there exists $v' \in class_{S'}^{L'}(v)$, such that the run of A' on v' ends in p . Hence, the run of A on $u' = \tilde{u} \cdot \langle \cdot v' \cdot \rangle$ ends in q , and furthermore, $u' \in class_S^L(u)$.

Subcase $\tilde{Q} \in dPrj^\Delta(D)$. Then, $Q = \tilde{Q}$ and $q \in acc^\Delta(\tilde{Q})$. Using the induction hypothesis applied to $\tilde{u}$, there exists $\tilde{u}' \in class_S^L(\tilde{u})$, such that $\langle \rangle^\Delta \xrightarrow{hdg(\tilde{u}')} q$. Since $\tilde{u}$ is irrelevant for L and S' , we have $\tilde{u}' \in class_S^L(u)$.

Case $u = \tilde{u} \cdot a$. This is easier to achieve than the previous cases and is thus omitted. $\square$

Lemma 16. A partial run of $A^{cgr(S)}$ assigns a state (Q, D) to a prefix $u \in \text{pref}(\mathcal{N}_\Sigma)$. If $Q \notin dPrj^\Delta(D)$ and $q \in Q$, then there exists a prefix $u' \in class_S^L(u)$, such that a partial run of A assigns state q to u' .

Proof. By induction on the length of u . In the base case, we have $u = \varepsilon$. The partial run of $A^{cgr(S)}$ on u assigns state (Q, D) , where $Q = I$ and $D = D_{init}^{A, F_S}$. Suppose that $Q \notin dPrj^\Delta(D)$ and $q \in Q$. Then, $I = \{q\}$. The unique partial run of A on $u = \varepsilon$ ends in the initial state $q \in I$. For the induction step, let $A' = A[I / \langle \rangle^\Delta]$, $L' = \mathcal{L}(A')$, and $S' = \mathcal{L}(A'[F / F_S])$. We distinguish the possible forms of u . So, $\tilde{u} \in \text{pref}(\mathcal{N}_\Sigma)$, $v \in \mathcal{N}_\Sigma$, and $a \in \Sigma$ exist, such that either of the following hold:

Case $u = \tilde{u} \cdot \langle \cdot v \cdot \rangle$. Let $(\tilde{Q}, \tilde{D})$ be the state in which the partial run of $A^{cgr(S)}$ on $\tilde{u}$ ends. From $Q \notin dPrj^\Delta(D)$, it follows that $\tilde{Q} \notin dPrj^\Delta(\tilde{D})$. Furthermore, $(\langle \rangle^\Delta, \tilde{D}) \xrightarrow{v} (Q, D)$ wrt $A^{cgr(S)}$. So, the partial run of $(A')^{cgr(S)}$ on v goes to state (Q, D) . Using the induction hypothesis, there exists $v' \in class_{S'}^{L'}(v)$, such that the partial run of A' on v' ends in q . Hence, the partial run of A on $u' = \tilde{u} \cdot \langle \cdot v' \cdot \rangle$ ends in q , and furthermore, $u' \in class_S^L(u)$.

Case $u = \tilde{u} \cdot \langle \cdot v \cdot \rangle$. Let $(\tilde{Q}, \tilde{D})$ be the state in which the partial run of $A^{cgr(S)}$ on $\tilde{u}$ ends. From $Q \notin dPrj^\Delta(D)$, it follows that $\tilde{Q} \notin dPrj^\Delta(\tilde{D})$. Furthermore, $(\langle \rangle^\Delta, \tilde{D}) \xrightarrow{v} (P, D)$ wrt $A^{cgr(S)}$ for some $P \subseteq Q$.

Subcase $P \in dPrj^\Delta(D)$. Then, $Q = \tilde{Q} @^\Delta acc^\Delta(P)$. So, there exists $\tilde{q} \in Q$ and $p \in acc^\Delta(P)$, such that $\tilde{q} @ p \rightarrow q$ in Δ . Using Lemma 15, there exists $v' \in class_{S'}^{L'}(v)$,

such that $\langle \rangle^\Delta \xrightarrow{hdg(v')} p$ wrt Δ . Using the induction hypothesis applied to $\tilde{u}$, there exists an initial state $q_0 \in I$ and a prefix $\tilde{u}' \in class_S^L(\tilde{u})$, such that the partial run of A on $\tilde{u}'$ goes on to state $\tilde{q}$. Hence, the partial run of A on $u' = \tilde{u}' \cdot \langle \cdot v' \cdot \rangle$ goes on to q and $u' \in class_S^L(u)$.

Subcase $P \notin dPrj^\Delta(D)$. Then, $Q = \tilde{Q} @ \Delta P$. So, there exists $\tilde{q} \in Q$ and $p \in P$, such that $\tilde{q} @ p \rightarrow q$ in Δ . Using Lemma 15, there exists $v' \in class_S^{L'}(v)$, such that $\langle \rangle^\Delta \xrightarrow{hdg(v')} p$ wrt Δ . Using the induction hypothesis applied to $\tilde{u}$, there exists an initial state $q_0 \in I$ and a prefix $\tilde{u}' \in class_S^L(\tilde{u})$, such that the partial run of A on $\tilde{u}'$ goes on to state $\tilde{q}$. Hence, the partial run of A on $u' = \tilde{u}' \cdot \langle \cdot v' \cdot \rangle$ goes on to q and $u' \in class_S^L(u)$.

Case $u = \tilde{u} \cdot a$. This is easier to achieve than the previous case and is thus omitted.

□

The next lemma states the key invariant of congruence projection, which eventually proves its completeness for subhedge projection. For this, we define for any $F \subseteq F_S \subseteq F$ the binary relation $\not\sim_{F_S}^F$ as the symmetric closure of $F \times (F_S \setminus F)$, i.e., for all $(q', q'') \in Q^2$:

$$q' \not\sim_{F_S}^F q'' \Leftrightarrow \left\{ \begin{array}{ll} (q' \in F & \wedge \quad q'' \in F_S \setminus F) \quad \vee \\ (q' \in F_S \setminus F & \wedge \quad q'' \in F) \end{array} \right.$$

Lemma 17 (Key Invariant). Let $A = (\Sigma, Q, \Delta, I, F)$ a dSHA, $F \subseteq F_S \subseteq Q$, $L = \mathcal{L}(A)$, and $S = \mathcal{L}(A[F/F_S])$. If $A^{gr(S)}$ has a partial run on prefix $u \in \text{pref}(\mathcal{N}_\Sigma)$ to state $(P, D) \in 2^Q \times \mathcal{D}^\Delta$, then for all $(p', p'') \in P^2$, $(r', r'') \in D$, and $(v', v'') \in \mathcal{N}_\Sigma^2$, such that

$$p' \xrightarrow{hdg(v')} r' \text{ wrt } \Delta \wedge p'' \xrightarrow{hdg(v'')} r'' \text{ wrt } \Delta$$

there exist $(u', u'') \in class_S^L(u)$, $w \in \text{suff}(\mathcal{N}_\Sigma)$, $q' \not\sim_{F_S}^F q''$, and $q_0 \in I$ such that

$$q_0 \xrightarrow{hdg(u' \cdot v' \cdot w)} q' \text{ wrt } \Delta \wedge q_0 \xrightarrow{hdg(u'' \cdot v'' \cdot w)} q'' \text{ wrt } \Delta$$

Proof. Induction on the number of dangling opening parenthesis of u .

In the base case, u does not have any dangling opening parenthesis so $u \in \mathcal{N}_\Sigma$ is a nested word. In this case, $D = D_{init}^{A, F_S}$. So, $A^{gr(S)}$ has a partial run on nested word $u \in \mathcal{N}_\Sigma$ to (P, D_{init}^{A, F_S}) , where $P \subseteq Q$. Let $(p', p'') \in P^2$, $(r', r'') \in D_{init}^{A, F_S}$, and $(v', v'') \in \mathcal{N}_\Sigma^2$, such that

$$p' \xrightarrow{hdg(v')} r' \text{ wrt } \Delta \wedge p'' \xrightarrow{hdg(v'')} r'' \text{ wrt } \Delta$$

It then follows that $P \notin dPrj^\Delta(D_{init}^{A, F_S})$. Therefore, we can apply Lemma 16. It shows that there exist nested words $(u', u'') \in (class_S^L(u))^2$ and $q_0 \in I$, such that

$$q_0 \xrightarrow{hdg(u')} p' \text{ wrt } \Delta \wedge q_0 \xrightarrow{hdg(u'')} p'' \text{ wrt } \Delta$$

Since $D_{init}^{A, F_S} \subseteq \text{ldr}^\Delta(F \times (F_S \setminus F))$ by Lemma 10, there exist a nested word $w \in \mathcal{N}_\Sigma$ and states $q' \not\sim_{F_S}^F q''$, such that

$$r' \xrightarrow{hdg(w)} q' \text{ wrt } \Delta \wedge r'' \xrightarrow{hdg(w)} q'' \text{ wrt } \Delta$$

Then, we have the following:

$$q_0 \xrightarrow{hdg(u' \cdot v' \cdot w)} q' \text{ wrt } \Delta \wedge q_0 \xrightarrow{hdg(u'' \cdot v'' \cdot w)} q'' \text{ wrt } \Delta$$

For the induction step, let $u = u_1 \cdot \langle \cdot u_2$ for some prefix $u_1 \in \text{pref}(\mathcal{N}_\Sigma)$ and nested word $u_2 \in \mathcal{N}_\Sigma$ so that u_1 has one open dangling bracket less than u . Let $A^{\text{cgr}(\mathbf{S})}$ have a partial run on nested word $u \in \mathcal{N}_\Sigma$ to (P, D) . Let (P_1, D_1) be the state that this run assigns to prefix u_1 . Then, $D = (D_1)_{P_1}$. Let $(p', p'') \in P^2$, $(r', r'') \in D$, and $(v', v'') \in \mathcal{N}_\Sigma^2$, such that

$$p' \xrightarrow{\text{hdg}(v')} r' \text{ wrt } \Delta \wedge p'' \xrightarrow{\text{hdg}(v'')} r'' \text{ wrt } \Delta$$

Since $D = (D_1)_{P_1}$, there exist $(p'_1, p''_1) \in (P_1)^2$ and states $(r'_1, r''_1) \in D_1$, such that $p'_1 @ q' \rightarrow r'_1$ and $p''_1 @ q'' \rightarrow r''_1$. In particular, $P_1 \notin d\text{Prj}^\Delta(D_1)$ so that we can apply Lemma 16. It shows that there exist $(u'_2, u''_2) \in \text{class}_{\mathbf{S}}^{L'}(u_2)$, such that $\langle \rangle^\Delta \xrightarrow{u'_2} p'$ and $\langle \rangle^\Delta \xrightarrow{u''_2} p''$. Let $v'_1 = \langle \cdot u'_2 \cdot v' \cdot \rangle$ and $v''_1 = \langle \cdot u''_2 \cdot v'' \cdot \rangle$. Then, $p'_1 \xrightarrow{\text{hdg}(v'_1)} r'_1$, $p''_1 \xrightarrow{\text{hdg}(v''_1)} r''_1$ wrt Δ . Using the induction hypothesis applied to u_1 , on which $A^{\text{cgr}(\mathbf{S})}$ has a partial run to state (P_1, D_1) , such that $(p'_1, p''_1) \in (P_1)^2$, $p'_1 \xrightarrow{\text{hdg}(v'_1)} r'_1$, $p''_1 \xrightarrow{\text{hdg}(v''_1)} r''_1$, and $(r'_1, r''_1) \in D$, there exist $(u'_1, u''_1) \in \text{class}_{\mathbf{S}}^L(u_1)$, $w_1 \in \text{suffs}(\mathcal{N}_\Sigma)$, $q' \not\approx_{F_{\mathbf{S}}}^F q''$, and $q_0 \in I$, such that

$$q_0 \xrightarrow{\text{hdg}(u'_1 \cdot v'_1 \cdot w_1)} q' \text{ wrt } \Delta \wedge q_0 \xrightarrow{\text{hdg}(u''_1 \cdot v''_1 \cdot w_1)} q'' \text{ wrt } \Delta$$

Let $u' = u'_1 \cdot \langle \cdot u'_2$, $u'' = u''_1 \cdot \langle \cdot u''_2$, and $w = \rangle \cdot w_1$. The above then yields the following:

$$q_0 \xrightarrow{\text{hdg}(u' \cdot v' \cdot w)} q' \text{ wrt } \Delta \wedge q_0 \xrightarrow{\text{hdg}(u'' \cdot v'' \cdot w)} q'' \text{ wrt } \Delta$$

Furthermore, $(u', u'') \in (\text{class}_{\mathbf{S}}^L(u))^2$, so this was to be shown. $\square$

Proposition 6. Let $A = (\Sigma, \mathcal{Q}, \Delta, I, F)$ a dSHA, $F \subseteq F_{\mathbf{S}} \subseteq \mathcal{Q}$, $L = \mathcal{L}(A)$, and $\mathbf{S} = \mathcal{L}(A[F/F_{\mathbf{S}}])$. If $A^{\text{cgr}(\mathbf{S})}$ has a partial run on prefix $u \in \text{pref}(\mathcal{N}_\Sigma)$ to some state $(P, D) \in 2^{\mathcal{Q}} \times \mathcal{D}^\Delta$ so that $P \notin d\text{Prj}^\Delta(D)$, then $\text{class}_{\mathbf{S}}^L(u)$ is subhedge relevant for L wrt $\mathbf{S}$.

Proof. Suppose that $A^{\text{cgr}(\mathbf{S})}$ has a partial run on prefix $u \in \text{pref}(\mathcal{N}_\Sigma)$ to some state $(P, D) \in 2^{\mathcal{Q}} \times \mathcal{D}^\Delta$ so that $P \notin d\text{Prj}^\Delta(D)$. Since $P \notin d\text{Prj}^\Delta(D)$, there exist $\text{acc}^\Delta(P)^2 \cap D \neq \text{emptyset}$. So, there exist $(p', p'') \in P^2$, $(r', r'') \in D$, and $(v', v'') \in \mathcal{N}_\Sigma^2$, such that

$$p' \xrightarrow{\text{hdg}(v')} r' \text{ wrt } \Delta \wedge p'' \xrightarrow{\text{hdg}(v'')} r'' \text{ wrt } \Delta$$

Using Lemma 17, there exist $(u', u'') \in \text{class}_{\mathbf{S}}^L(u)$, $w \in \text{suffs}(\mathcal{N}_\Sigma)$, $q' \not\approx_{F_{\mathbf{S}}}^F q''$, and $q_0 \in I$, such that

$$q_0 \xrightarrow{\text{hdg}(u' \cdot v' \cdot w)} q' \text{ wrt } \Delta \wedge q_0 \xrightarrow{\text{hdg}(u'' \cdot v'' \cdot w)} q'' \text{ wrt } \Delta$$

Hence, $\text{class}_{\mathbf{S}}^L(u)$ is relevant for L wrt $\mathbf{S}$. $\square$

Theorem 3 (Completeness of congruence projection.). For any complete dSHA $(\Sigma, \mathcal{Q}, \Delta, I, F)$ and set $F \subseteq F_{\mathbf{S}} \subseteq \mathcal{Q}$, the congruence projection $A^{\text{cgr}(\mathbf{S})}$ is sound and complete for subhedge projection for the regular pattern $\mathcal{L}(A)$ wrt the regular schema $\mathbf{S} = \mathcal{L}(A[F_{\mathbf{S}}/F])$.

Proof. The soundness of congruence projection is shown in Theorem 2. For proving completeness, let $u \in \text{pref}(\mathcal{N}_\Sigma)$ be a nested word prefix that is strongly subhedge irrelevant for L wrt $\mathbf{S}$. By definition of strong irrelevance, the class $\text{class}_{\mathbf{S}}^L(u)$ is irrelevant for L and $\mathbf{S}$. Let (P, D) be the unique state assigned by the partial run of $A^{\text{cgr}(\mathbf{S})}$ to prefix u . Since $\text{class}_{\mathbf{S}}^L(u)$ is irrelevant for L and $\mathbf{S}$, Proposition 6 shows that $P \in d\text{Prj}^\Delta(D)$. Using Lemma 14, (P, D) is then a subhedge projection state of $A^{\text{cgr}(\mathbf{S})}$. $\square$

7.5. In-Memory Complexity

We next discuss the complexity of membership testing with complete subhedge projection based on the in-memory evaluation of the input hedge by the congruence projection of the input automaton.

Lemma 18. *The number of states $|Q^{cgr}|$ is in $O(2^{n^2+n})$, where n is the number of states of A .*

Proof. With the deterministic construction, the states of $A^{cgr(S)}$ are pairs in $2^Q \times \mathcal{D}^\Delta$. So, the maximal number of states of the congruence projection is $|Q^{cgr(S)}| = 2^{|Q|} |\mathcal{D}^\Delta| \leq 2^n 2^{n^2} = 2^{n^2+n}$. $\square$

Let $\text{diff-rel}(Q^{cgr(S)})$ be the set of difference relations used in the states of $A^{cgr(S)}$.

Corollary 1. *Let the signature Σ be fixed. For any complete dSHA A with schema $S \subseteq \mathcal{H}_\Sigma$, the membership of any hedge $h \in S$ can be tested in memory in time $O(|1|)$ per nonprojected node of h after a preprocessing time of $O(|A^{cgr(S)}| + n^3 d)$, where d is the cardinality of $\text{diff-rel}(Q^{cgr(S)})$ and n is the number of states of A .*

Since $|Q^{cgr(S)}|$ is in $O(2^{n^2+n})$ by Lemma 18, the preprocessing time $O(|A^{cgr(S)}| + n^3 d)$ is in $O(2^{2n^2+2n})$, too.

Proof. Since A is complete, Theorem 3 shows that $A^{cgr(S)}$ is complete for subhedge projection for schema S . The in-memory evaluator of $A^{cgr(S)}$ can be run on any input hedge $h \in S$ in time $O(|1|)$ per nonprojected node of h after a preprocessing time of $O(|A^{cgr(S)}| + n^3 d)$, where d is the cardinality of $\text{diff-rel}(A^{cgr(S)})$. Since the signature Σ is fixed and $A^{cgr(S)}$ deterministic, $|A^{cgr(S)}|$ is in $O(|Q^{cgr(S)}|^2)$. The dominating part $|A^{cgr(S)}|$ is in $O(|Q^{cgr(S)}|^2)$. $\square$

When applied to regular XPath query answering as in our experiments in Section 11, we have $n \leq 80$ and $d \leq 25$ so the cubic part $O(n^3 d)$ is not limiting. The sizes of the SHA[↓]s are below 2600 counting both states and rules. The upper bound $O(|A^{cgr(S)}| + n^3 d)$ thus suggests that the preprocessing time may be feasible in practice, even though exponential in the worst case.

We must be careful here since the sizes of the SHA[↓]s reported in Section 11 do not refer to the pure congruence projection $A^{cgr(S)}$ but to the earliest congruence projection $A_{e(S)}^{cgr(S)}$ that we will introduce only in Section 8. The noncreation of transition rules to states that are safe for rejection may lead to a radical size reduction. Alternatively, an important size reduction in $A^{cgr(S)}$ could be obtained by removing all useless states and transition rules. But, this would not reduce the precomputation time, just the opposite: it would increase the preprocessing time since cleaning automata a posteriori also needs time, which for SHA[↓]s may be nonlinear.

Instead of precomputing $A^{cgr(S)}$ from the inputs A and F_S statically, we can compute only the part needed of $A^{cgr(S)}$ for evaluating the input hedge h on-the-fly. We note that the number of transition rules of the needed part is bounded by $O(|h|)$ but may be way smaller due to projection. The preprocessing time then goes down to $O(|A|)$. Since the full automaton $A^{cgr(S)}$ may be exponential in n in the worst case, this may reduce the overall processing time.

On the positive side, on-the-fly evaluation reduces the overall computation time to $O(|h| + n^3 d(h))$, where $d(h)$ is the number of difference relations in the part of $A^{cgr(S)}$ needed for the evaluation of h , so $d(h) \leq \min(|h|, d)$. On the down side, $d(h)$ steps will no more be in constant but cubic time in $O(n^3)$. It should also be noted that the overall time of the safe-no-change projection on-the-fly evaluation is in $O(|A| + |h| + n s(h))$, where $s(h)$ is the number of subsets of safe states of A in states of Q^{cgr} , i.e., $s(h) \leq |h|$. This is since each subset of safe states can be computed in linear time in $O(n)$. As argued above, the overall time of the congruence projection obtained by on-the-fly evaluation is in $O(|A| + |h| + n^3 d(h))$. This is because the computation of a difference relation is in $O(n^3)$.

by using ground Datalog programs of cubic size. The additional cubic factor in $n^3 d(h)$ instead of a linear factor in $n s(h)$ is the price to be paid for achieving complete subhedge projection with on-the-fly evaluation. Note that this price is largely acceptable in cases where $n \leq 80$, $d(h) \leq 15$, and $s(h) \leq 15$.

Example 15 (Exponential blow-up.). *In order to see how the exponential worst case may happen, we consider a family of regular languages, for which the minimal left-to-right DFA is exponentially bigger than the minimal right-to-left DFA. The classical example languages with this property are $L_n = \Sigma^* . a . \Sigma^n$, where $n \in \mathbb{N}$ and $\Sigma = \{a, b\}$. Intuitively, a word in Σ^* belongs to L_n if and only if its $n + 1$ -th letter from the end is equal to "a". The minimal left-to-right DFA for L_n has 2^{n+1} many states since it needs to memorize a window of $n + 1$ letters. In contrast, its minimal right-to-left DFA has only $n + 1$ states; in this direction, it is sufficient to memorize the distance from the end modulo $n + 1$.*

We next consider the schema S defined by $\mu z. \langle (a + b) \cdot z \rangle^*$. It contains all hedges having in all its subtrees a label from $\Sigma = \{a, b\}$ as the first letter, and no further occurrences of letters of Σ elsewhere. We then consider the family of hedge languages $H_n \in \mathcal{H}_\Sigma$ with schema S , such that the sequence of labels of some root-to-leaf path of h_n belongs to L_n . Note that H_n can be recognized in a bottom-up manner by the dSHA A_n with $O(n + 1)$ states, which simulates the minimal deterministic DFA of L_n on all paths of the input hedge. For an evaluator with subhedge projection, the situation is different. When moving top-down, it needs to memorize the sequence of labels of the $n + 1$ -last ancestors, possibly filled with b 's, and there are 2^{n+1} of such sequences. If, for some leaf, its sequence starts with an "a" then the following subhedges with the following leaves can be projected away. As a consequence, there cannot be any $dSHA^\downarrow$ recognizing H_n that projects away all irrelevant subhedges with less than 2^{n+1} states. In particular, the size of $A_n^{cgr(S)}$ must be exponential in the size of A_n .

8. Earliest Membership with Subhedge Projection

We next enhance our compilers for introducing subhedge projection states, such that they can also detect certain membership or nonmembership in an earliest manner. For this, we show how to combine the previous earliest membership tester for dSHAs from [21] with any subtree projection algorithm so that it applies to both, to safe-no-change projection and to congruence projection. By combining both aspects orthogonally, we obtain an earliest dSHA with subhedge projection, which can be evaluated either in memory or in streaming mode.

8.1. Earliest Membership

We start with a semantic characterization of earliest membership, i.e., of prefixes of nested words of hedges whose suffixes are irrelevant for membership, when assuming top-down and left-to-right processing. Note that the same characterization up to various presentation choices was given earlier in the context of stream processing [21,22].

Definition 13. *Let $S \subseteq \mathcal{H}_\Sigma$ be a hedge schema and $L \subseteq S$ a hedge language satisfying this schema. A nested word prefix v is certain for membership in L with schema S for all suffixes w of nested words such that $v \cdot w \in nw(S)$ it holds that $v \cdot w \in nw(L)$. It is called certain for nonmembership in L with schema S for all suffixes w of nested words such that $v \cdot w \in nw(S)$ it holds that $v \cdot w \notin nw(L)$.*

We are now interested in $SHA^\downarrow$ that are able to detect certain membership and nonmembership syntactically at the earliest possible prefix or node of the run.

Definition 14. *A $dSHA^\downarrow A = (\Sigma, Q, \Delta, I, F)$ with schema $S \subseteq \mathcal{H}_\Sigma$ is called earliest for schema S if there exists a state $sel \in Q$, such that for all hedges $h \in S$, the unique run R of $compl(A)$ on h satisfies the following:*

- it goes to the state *sel* for exactly all those prefixes of $nw(h)$ that are certain for membership in $\mathcal{L}(A)$ wrt S ;
- it goes to the sink for all prefixes of $nw(h)$ that are certain for nonmembership in $\mathcal{L}(A)$ with S .

For dSHA A , schema-complete for schema $S = \mathcal{L}(A[F/F_S])$, we can construct a dSHA $^\downarrow$ $A_{e(S)}$ that is the earliest for S , as shown in Appendix A. The idea is to adapt the earliest membership tester for deterministic SHAs from [21] so that it takes schema restrictions into account. Furthermore, we need to compile the input dSHAs to dSHA $^\downarrow$ s rather than to dNWAs, as used there. This change is straightforward once the relationship between both automata models becomes clear.

Given that the construction of $A_{e(S)}$ is not part of the core of the present paper, we only illustrate its outcome at our running example: for the dSHA A with schema $S = \llbracket \text{doc} \rrbracket$, in Figure 4, we obtain the SHA $^\downarrow$ $A_{e(S)}$ in Figure 19. Note that, on the prefix $\langle \text{list} \langle \text{item}$, it goes to state *sel*, showing that it is certain for membership. On prefix $\langle \text{item}$, it blocks showing that it is certain for nonmembership. However, on the prefix $\langle \text{list} \langle \text{list}$, it does not go into a subhedge projection state even though this prefix is subhedge irrelevant.

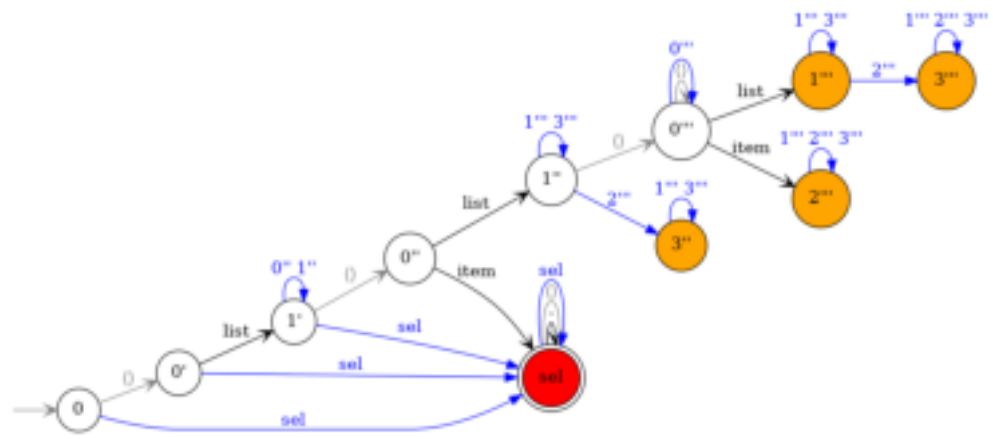


Figure 19. The earliest dSHA $^\downarrow$ $A_{e(S)}$ for the dSHA A in Figure 4 with schema $S = \llbracket \text{doc} \rrbracket$ for the XPath filter $[\text{self}::\text{list}/\text{child}::\text{item}]$.

8.2. Adding Subhedge Projection

We now enhance any earliest membership for dSHAs with subhedge projection. The idea is to combine subhedge projection with earliest membership. So, suppose that we are given a schema S and two automata, A^π and A_e , with schema S that recognize the same language up to the schema, so $\mathcal{L}(A^\pi) \cap \mathcal{L}(S) = \mathcal{L}(A_e) \cap \mathcal{L}(S)$. Furthermore, assume that A_e has a selection state *sel*, indicating certain membership at the earliest possible prefix.

In order to obtain the earliest membership with subhedge projection, we combine the two SHA $^\downarrow$ s into a single SHA $^\downarrow$, $A_e^\pi = (\Sigma, \mathcal{Q}_e^\pi, \Delta_e^\pi, I_e^\pi, F_e^\pi)$, basically running both automata in parallel but under shared control. The state set of A_e^π are as follows:

$$\begin{aligned} \mathcal{Q}_e^\pi &= (\mathcal{Q}^\pi \times (\mathcal{Q}_e \setminus \{\text{sel}\})) \cup \{\text{sel}\} \\ I_e^\pi &= (I^\pi \times (I_e \setminus \{\text{sel}\})) \cup \{\text{sel} \mid \text{sel} \in I_e\} \\ F_e^\pi &= (F^\pi \times (F_e \setminus \{\text{sel}\})) \cup \{\text{sel}\} \end{aligned}$$

The transition rules in Δ_e^π are given in Figure 20. Any run of A_e^π synchronizes parallel runs of A^π and A_e as follows: whenever A_e goes to *sel*, then A_e^π does so too, and whenever A^π goes into a subhedge projection state, then it makes A_e jump over the subsequent subhedge.

$$\begin{array}{c}
\frac{q \xrightarrow{a} q' \text{ in } \Delta^\pi \quad r \xrightarrow{a} r' \text{ in } \Delta_e \quad q \notin P \quad r' \neq \text{sel}}{(q, r) \xrightarrow{a} (q', r') \text{ in } \Delta_e^\pi} \quad \frac{q \xrightarrow{a} q' \text{ in } \Delta^\pi \quad r \xrightarrow{a} \text{sel} \text{ in } \Delta_e}{(q, r) \xrightarrow{a} \text{sel} \text{ in } \Delta_e^\pi} \\
\\
\frac{q \xrightarrow{a} q' \text{ in } \Delta^\pi \quad r \xrightarrow{a} r' \text{ in } \Delta_e \quad q \in P \quad r' \neq \text{sel}}{(q, r) \xrightarrow{a} (q', r) \text{ in } \Delta_e^\pi} \\
\\
\frac{q@p \rightarrow q' \text{ in } \Delta^\pi \quad r@s \rightarrow r' \text{ in } \Delta_e \quad r' \neq \text{sel} \quad q \notin P}{(q, r)@(p, s) \rightarrow (q', r') \text{ in } \Delta_e^\pi} \quad \frac{q@p \rightarrow q' \text{ in } \Delta^\pi \quad r@s \rightarrow \text{sel} \text{ in } \Delta_e}{(q, r)@(p, s) \rightarrow \text{sel} \text{ in } \Delta_e^\pi} \\
\\
\frac{q@p \rightarrow q' \text{ in } \Delta^\pi \quad r@s \rightarrow r' \text{ in } \Delta_e \quad q \in P \quad r' \neq \text{sel}}{(q, r)@(p, s) \rightarrow (q', r) \text{ in } \Delta_e^\pi} \\
\\
\frac{q \xrightarrow{\langle \rangle} q' \text{ in } \Delta^\pi \quad r \xrightarrow{\langle \rangle} r' \text{ in } \Delta_e \quad q \notin P \quad r' \neq \text{sel}}{(q, r) \xrightarrow{\langle \rangle} (q', r') \text{ in } \Delta_e^\pi} \quad \frac{q \xrightarrow{\langle \rangle} q' \text{ in } \Delta^\pi \quad r \xrightarrow{\langle \rangle} \text{sel} \text{ in } \Delta_e}{(q, r) \xrightarrow{\langle \rangle} \text{sel} \text{ in } \Delta_e^\pi} \\
\\
\frac{q \xrightarrow{\langle \rangle} q' \text{ in } \Delta^\pi \quad r \xrightarrow{\langle \rangle} r' \text{ in } \Delta_e \quad q \in P \quad r' \neq \text{sel}}{(q, r) \xrightarrow{\langle \rangle} (q', r) \text{ in } \Delta_e^\pi} \\
\\
\frac{a \in \Sigma}{\text{sel} \xrightarrow{a} \text{sel} \text{ in } \Delta_e^\pi} \quad \frac{\mu \in \mathcal{Q}_e^\pi}{\text{sel}@\mu \rightarrow \text{sel} \text{ in } \Delta_e^\pi} \quad \frac{\mu \in \mathcal{Q}_e^\pi}{\mu@\text{sel} \rightarrow \text{sel} \text{ in } \Delta_e^\pi} \quad \frac{\text{true}}{\text{sel} \xrightarrow{\langle \rangle} \text{sel} \text{ in } \Delta_e^\pi}
\end{array}$$

Figure 20. The transition rules Δ_e^π inferred from those of the $\text{SHA}^\downarrow$ s A^π with projection states P and A_e with selection state sel .

For illustration, we reconsider the dSHA from Figure 4. The earliest dSHA[↓] with subhedge projection $A_{e(S)}^{\text{Snc}}$ is given in Figure 21. Here, we combine the safe-no-change projection A^{Snc} from Figure 11 with the earliest membership tester $A_{e(S)}$ from Figure 19. Their combination $A_{e(S)}^{\text{Snc}}$ represents the pattern `[self::list/child::item]` with schema `[[doc]]` in a concise manner. After prefix `<list<item`, it detects certain membership: for the prefix `<list<list`, it detects that the subhedge is irrelevant, and after prefix `<item`, it detects certain nonmembership.

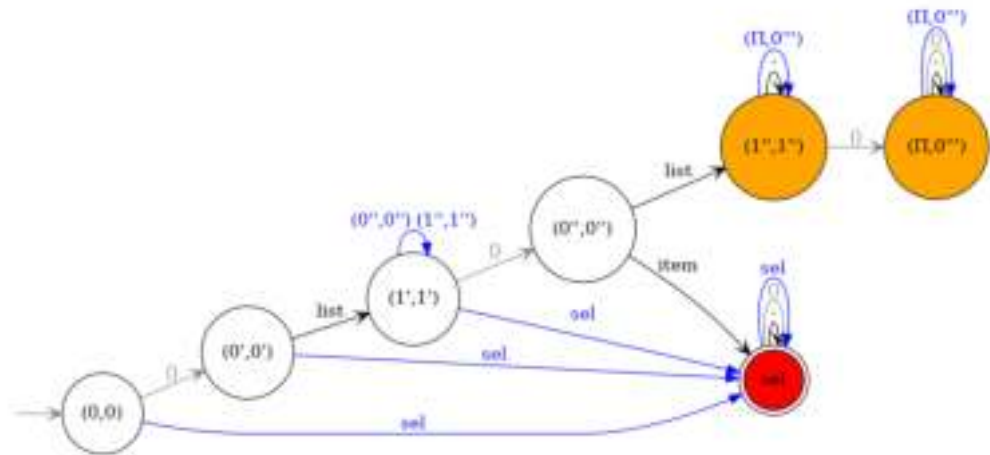


Figure 21. The earliest dSHA[↓] $A_{e(S)}^{\text{Snc}}$ with safe-no-change subhedge projection for the dSHA A in Figure 4 with schema $S = \llbracket \text{doc} \rrbracket$ for the XPath filter `[self::list/child::item]`.

8.3. Soundness and Completeness

We next show that the soundness and completeness of a projection algorithm are preserved when combined with some earliest membership tester when assuming determinism.

Proposition 7. *Let A^π and A_e be $d\text{SHA}^\downarrow$ s with the same schema $\mathbf{S}$ and the same language up to the schema, i.e., $\mathcal{L}(A^\pi) \cap \mathcal{L}(\mathbf{S}) = \mathcal{L}(A_e) \cap \mathcal{L}(\mathbf{S})$. The combination $d\text{SHA}^\downarrow A_e^\pi$ then has the same language up to schema $\mathbf{S}$, too. Furthermore, if A_e is earliest for $\mathbf{S}$, then A_e^π is earliest for $\mathbf{S}$ and goes into some subhedge projection state whenever A^π does.*

Proof. If q is a subhedge projection state of A^π and r a state of A_e , then (q, r) is a subhedge projection state of A_e^π . The evaluator for automaton A_e^π runs A^π and A_e in parallel on the input hedge, while skipping subhedges starting in subhedge projection states of A^π . These include the subhedges starting in a projection state when evaluating A_e^π on h . Applying Proposition 2 to A^π , such subhedges are irrelevant for $\mathcal{L}(A^\pi)$ with respect to $\mathbf{S}$, since A^π is assumed to be deterministic. Since $\mathcal{L}(A^\pi) \cap \mathbf{S} = \mathcal{L}(A_e) \cap \mathbf{S}$, skipping such a subhedge does not affect acceptance by A_e for hedges inside schema $\mathbf{S}$. Therefore, the evaluator of A_e^π is earliest with respect to $\mathbf{S}$, too. $\square$

The above proposition shows that if A^π is complete for subhedge projection, then A_e^π is also complete for subhedge projection while also being earliest.

Theorem 4. *For any complete $d\text{SHA}$ A with schema $\mathbf{S}$, the $d\text{SHA } A^{cgr}(\mathbf{S})_{e(\mathbf{S})}$ is earliest and complete for subhedge projection for schema $\mathbf{S}$.*

Proof. The congruence projection $A^{cgr}(\mathbf{S})$ is sound by Proposition 2 and complete for subhedge projection wrt $\mathbf{S}$ by Theorem 3. The automaton $A_{e(\mathbf{S})}$ discussed in Section 8.1 is earliest for $\mathbf{S}$. So, their combination $A^{cgr}(\mathbf{S})_{e(\mathbf{S})}$ yields the earliest automaton with complete subhedge projection wrt $\mathbf{S}$ using Proposition 7. $\square$

8.4. In-Memory Complexity

We next discuss the complexity of earliest membership testing with complete subhedge projection by an in-memory evaluator for $A^{cgr}(\mathbf{S})_{e(\mathbf{S})}$.

Lemma 19. *The number of states in $Q^{cgr}(\mathbf{S})_{e(\mathbf{S})}$ is in $O(2^{n^2+2n+\log(n)})$, where $n = |\mathcal{Q}|$.*

Proof. Using Lemma 18, the number of states of $A^{cgr}(\mathbf{S})$ is in $O(2^{n^2+n})$, where $n = |\mathcal{Q}|$. The number of states of $A_{e(\mathbf{S})}$ is in $O(n2^n)$, and thus, in $O(2^{n+\log(n)})$. The number of states of $Q^{cgr}(\mathbf{S})_{e(\mathbf{S})}$ is thus in $O(2^{n^2+n} 2^{n+\log(n)})$, which is in $O(2^{n^2+2n+\log(n)})$. $\square$

Corollary 2. *Let the signature Σ be fixed. For any complete $d\text{SHA}$ A with schema $\mathbf{S} \subseteq \mathcal{H}_\Sigma$, earliest membership on a input hedge $h \in \mathbf{S}$ can be tested in memory in time $O(|1|)$ per nonprojected node of h after a preprocessing time of $O(|A^{cgr}(\mathbf{S})_{e(\mathbf{S})}| + n^3d + ns)$, where d is the number of difference relations and s is the number of subset of safe states in $Q^{cgr}(\mathbf{S})_{e(\mathbf{S})}$. The preprocessing time is also in $O(2^{2n^2+4n+2\log(n)})$, where n is the number of states of A .*

Proof. Since A is complete, Theorem 4 shows that $A^{cgr}(\mathbf{S})_{e(\mathbf{S})}$ is the earliest, sound, and complete for subhedge projection for schema $\mathbf{S}$. The in-memory evaluator of $A^{cgr}(\mathbf{S})_{e(\mathbf{S})}$ can be run on any input hedge $h \in \mathbf{S}$ in time $O(|1|)$ per nonprojected node of h after a preprocessing time of $O(|A^{cgr}(\mathbf{S})_{e(\mathbf{S})}| + n^3d + ns)$. Σ is fixed to the size $|A^{cgr}(\mathbf{S})_{e(\mathbf{S})}|$ bounded by $O(|Q^{cgr}(\mathbf{S})_{e(\mathbf{S})}|^2)$. Lemma 19 shows that $|Q^{cgr}(\mathbf{S})_{e(\mathbf{S})}|$ is in $O(2^{n^2+2n+\log(n)})$, so $|A^{cgr}(\mathbf{S})_{e(\mathbf{S})}|$ is in $O(2^{2n^2+4n+2\log(n)})$. $\square$

As before, instead of precomputing $A^{egr}(\mathbf{S})_{e(\mathbf{S})}$ from the input dSHA A and F_S , we can compute only the part needed for evaluating the input hedge h on-the-fly. If the needed part of $A^{egr}(\mathbf{S})_{e(\mathbf{S})}$ for evaluating h is small, the overall time and space go down considerably.

9. Streaming Algorithms

We show that $\text{dSHA}^\downarrow$ s can be evaluated on the nested word of a hedge in a streaming manner, so that they produce the same results and projection behavior as by evaluating them on hedges in-memory in a top-down and left-to-right manner.

All aspects related to subhedge projection remain unchanged. Earliest query answering in streaming mode means that nested word suffixes are ignored if irrelevant, and not only nested words as with subhedge projection. What changes between both evaluation modes is memory consumption. For in-memory evaluation, the whole graph of the input hedge must be stored, while in streaming mode, only a stack and a state are maintained at any event. For testing regular language membership, the state space is finite, while the stack space become infinite, since the depth of the stack depends on the depth of the input hedge.

The property of having equivalent streaming and in-memory evaluators was already noticed in [26] for Neumann and Seidl's pushdown forest automata [25] and for nested word automata [11,15]. We here show how to transpose this result to $\text{dSHA}^\downarrow$ s. This permits us to show the correctness of our evaluation algorithms based on properties of the constructed automata. It also permits obtaining correctness properties of streaming algorithms by reasoning about the in-memory evaluator of $\text{SHA}^\downarrow$ s.

9.1. Streaming Evaluators for Downward Hedge Automata

We define the streaming evaluator of a $\text{SHA}^\downarrow A = (\Sigma, \mathcal{Q}, \Delta, I, F)$ by a visibly pushdown machine. The set of configurations of this machine is $\mathcal{K} = \mathcal{Q} \times \mathcal{Q}^*$, containing pairs of states and stacks of states. The visibly pushdown machine provides for any word $v \in \hat{\Sigma}^*$ a streaming transition relation $\xrightarrow{v \text{ str}} \subseteq \mathcal{K} \times \mathcal{K}$, such that for all states $q, q' \in \mathcal{Q}$ and stacks $\sigma \in \mathcal{Q}^*$ and words $v, v' \in \hat{\Sigma}^*$:

$$\begin{array}{c} \frac{\text{true}}{(q, \sigma) \xrightarrow{\varepsilon \text{ str}} (q, \sigma) \text{ wrt } \Delta} \quad \frac{(q, \sigma) \xrightarrow{v \text{ str}} (q', \sigma) \quad (q', \sigma) \xrightarrow{v' \text{ str}} (q'', \sigma) \text{ wrt } \Delta}{(q, \sigma) \xrightarrow{v \cdot v' \text{ str}} (q'', \sigma) \text{ wrt } \Delta} \\[10pt] \frac{q \xrightarrow{a} q' \text{ in } \Delta}{q \xrightarrow{a \text{ str}} q' \text{ wrt } \Delta} \quad \frac{q \xrightarrow{\langle \rangle} q' \text{ in } \Delta}{(q, \sigma) \xrightarrow{\langle \text{ str } \rangle} (q', \sigma \cdot q) \text{ wrt } \Delta} \quad \frac{q @ p \rightarrow q' \text{ in } \Delta}{(p, \sigma \cdot q) \xrightarrow{\rangle \text{ str } \rangle} (q', \sigma) \text{ wrt } \Delta} \end{array}$$

We note that the same visibly pushdown machine can be obtained by compiling the $\text{SHA}^\downarrow$ to an NWA, whose streaming evaluator is defined by a visibly pushdown machine too (see Section 12.3 on related work).

The notion of partial runs of the in-memory evaluator is closely related to streaming runs. To see this, we first note that each partial run $v \text{ wrt } \Delta$ naturally defines a stack as follows. Since v is a nested word prefix, there exist unique nested words $v_0, \dots, v_n \in \text{nw}(\mathcal{H}_{\Sigma \cup \mathcal{Q}})$, such that

$$v = v_0 \cdot \langle \cdot v_1 \cdot \langle \cdot \dots \cdot \langle \cdot v_n \cdot$$

These nested words must be partial runs of Δ too, so there exist states $p_0, \dots, p_n, q_0, \dots, q_n \in \mathcal{Q}$, such that v_i goes from p_i to q_i for all $0 \leq i \leq n$. We define the stack σ of v by $\sigma = q_0 \cdot \dots \cdot q_{n-1}$ and say that v goes from p_0 to q_n and has stack σ . Note that the stack is empty if v is a nested word, since in this case $n = 0$ so that $v = v_0$ and $\sigma = \varepsilon$. For instance, the partial run $v_0 = 0 \cdot \langle \cdot 0 \cdot \text{list} \cdot 1 \cdot \langle \cdot 0 \cdot \text{item} \cdot 2$ has the stack $\sigma_0 = 0 \cdot 1$ while the partial run $v_1 = v_0 \cdot \rangle \cdot 3 \cdot \rangle \cdot 4$ has stack $\sigma_1 = \varepsilon$.

Lemma 20. Consider a SHA $A = (\Sigma, \mathcal{Q}, \Delta, I, F)$, states $q, q' \in \mathcal{Q}$, and a stack $\sigma \in \mathcal{Q}^*$. Let v be a prefix of some nested word in $nw(\mathcal{H}_\Sigma)$. Then, there exists a partial run r wrt Δ from q to q' with stack σ such that $\text{proj}_\Sigma(r) = v$ iff $(q, \varepsilon) \xrightarrow{v \text{ str}} (q', \sigma)$ wrt Δ .

Proof. This is carried out by induction on the length of prefix v . $\square$

This lemma implies that any in-memory transition of A on a hedge h from q to q' can be identified with some streaming transition of A on $nw(h)$ from (q, ε) to (q', ε) .

Proposition 8. $q \xrightarrow{h} q'$ wrt $\Delta \Leftrightarrow (q, \varepsilon) \xrightarrow{nw(h) \text{ str}} (q', \varepsilon)$ wrt Δ .

Proof. If $q \xrightarrow{h} q'$ wrt Δ , there exists a run R of h with Δ that starts with q and ends with q' . Consider the partial run $v = nw(R)$ on h . Since v is a nested word, its stack is empty. By Lemma 20, we have $(q, \varepsilon) \xrightarrow{\text{proj}_\Sigma(v) \text{ str}} (q', \varepsilon)$ wrt Δ and $\text{proj}_\Sigma(v) = nw(h)$. For the inverse implication, assume that $(q, \varepsilon) \xrightarrow{nw(h) \text{ str}} (q', \varepsilon)$ wrt Δ . Let $u = nw(h)$. By Lemma 20, there exists a partial run v of Δ from q to q' with stack ε . Then, there exists a run R of Δ on h such that $v = nw(R)$. Hence, $q \xrightarrow{h} q'$ wrt Δ . $\square$

For any deterministic $\text{dSHA}^\downarrow A$, hedge h , and state q , the streaming transition on $nw(h)$ with respect to Δ starting with (q, ε) can be computed in time $O(1)$ per letter after a precomputation in time $O(|A|)$. So, the overall computation time is in $O(|A| + |h|)$. The streaming memory needed to store a configuration is of size $O(\text{depth}(h) + |A|)$ since the size of the visible stack is bounded by the depth of the input hedge.

9.2. Adding Subhedge Projection

We next extend the streaming transition relation with subhedge projection, in analogy to what we did for the in-memory transition relation. We define a transition relation with subhedge projection $\xrightarrow{h \text{ str}}_{shp} \subseteq \mathcal{K} \times \mathcal{K}$ with respect to Δ , such that for all words $v, v' \in \hat{\Sigma}^*$, letters $a \in \Sigma$, states $p, q, q', q'' \in \mathcal{Q}$, and stacks $\sigma, \sigma', \sigma'' \in \mathcal{Q}^*$:

$$\begin{array}{c}
 \frac{q \in \mathcal{Q}_{shp}^\Delta \quad v \in nw(\mathcal{H}_\Sigma)}{(q, \sigma) \xrightarrow{v \text{ str}}_{shp} (q, \sigma) \text{ wrt } \Delta} \\
 \\
 \frac{q \notin \mathcal{Q}_{shp}^\Delta \quad q \xrightarrow{a} q' \text{ in } \Delta}{(q, \sigma) \xrightarrow{a \text{ str}}_{shp} (q', \sigma) \text{ wrt } \Delta} \quad \frac{q \notin \mathcal{Q}_{shp}^\Delta}{(q, \sigma) \xrightarrow{\varepsilon \text{ str}}_{shp} (q, \sigma) \text{ wrt } \Delta} \\
 \\
 \frac{q \notin \mathcal{Q}_{shp}^\Delta \quad (q, \sigma) \xrightarrow{v \text{ str}}_{shp} (q', \sigma') \text{ wrt } \Delta \quad (q', \sigma') \xrightarrow{v' \text{ str}}_{shp} (q'', \sigma'') \text{ wrt } \Delta}{(q, \sigma) \xrightarrow{v \cdot v' \text{ str}}_{shp} (q'', \sigma'') \text{ wrt } \Delta} \\
 \\
 \frac{q \notin \mathcal{Q}_{shp}^\Delta \quad q \xrightarrow{\langle \rangle} q' \text{ in } \Delta}{(q, \sigma) \xrightarrow{\langle \rangle \text{ str}}_{shp} (q', \sigma \cdot q) \text{ wrt } \Delta} \quad \frac{p \notin \mathcal{Q}_{shp}^\Delta \quad q @ p \rightarrow q' \text{ in } \Delta}{(p, \sigma \cdot q) \xrightarrow{\rangle \text{ str}}_{shp} (q', \sigma) \text{ wrt } \Delta}
 \end{array}$$

The projecting transition relation stays in a configuration with a subhedge projection state until the end of the current subhedge is reached. This is correct since a subhedge projection state cannot be changed anyway.

Proposition 9. For any $\text{SHA}^\downarrow A = (\Sigma, \mathcal{Q}, \Delta, I, F)$, states $q, q' \in \mathcal{Q}$, nested word v on which Δ has no blocking partial run starting from q :

$$(q, \varepsilon) \xrightarrow{v}^{str} (q', \varepsilon) \text{ wrt } \Delta \Leftrightarrow (q, \varepsilon) \xrightarrow{v}_{shp}^{str} (q', \varepsilon) \text{ wrt } \Delta.$$

Proof. Let $h \in \mathcal{H}_\Sigma$ be the hedge such that $v = nw(h)$. The proof is by induction on the size of v . For the implication from the left to the right we assume that $(q, \varepsilon) \xrightarrow{v}^{str} (q', \varepsilon) \text{ wrt } \Delta$. Since $v = nw(h)$, Proposition 8 proves $q \xrightarrow{h}^{str} q' \text{ wrt } \Delta$.

Case $q \in \mathcal{Q}_{shp}^\Delta$. By Lemma 6, it follows from $q \xrightarrow{h}^{str} q' \text{ wrt } \Delta$ that $q = q'$. We can then apply the following rule since v is a nested word:

$$\frac{q \in \mathcal{Q}_{shp}^\Delta \quad v = nw(\mathcal{H}_\Sigma)}{(q, \varepsilon) \xrightarrow{v}_{shp}^{str} (q, \varepsilon) \text{ wrt } \Delta}$$

$$\text{Hence, } (q, \varepsilon) \xrightarrow{v}_{shp}^{str} (q, \varepsilon) \text{ wrt } \Delta.$$

Case $q \notin \mathcal{Q}_{shp}^\Delta$. We distinguish all possible forms of hedge h .

Subcase $h = h' \cdot h''$. Let $v' = nw(h')$ and $v'' = nw(h'')$ so $v = v' \cdot v'' = nw(h)$. The following rule must have been applied:

$$\frac{(q, \varepsilon) \xrightarrow{v'}^{str} (p, \sigma) \text{ wrt } \Delta \quad (p, \sigma) \xrightarrow{v''}^{str} (q', \varepsilon) \text{ wrt } \Delta}{(q, \varepsilon) \xrightarrow{v' \cdot v''}^{str} (q', \varepsilon) \text{ wrt } \Delta}$$

Since v' is a nested word, it follows that $\sigma = \varepsilon$. By the induction hypothesis applied to the nested words v' and v'' , we get $(q, \varepsilon) \xrightarrow{v'}^{str}_{shp} (p, \varepsilon) \text{ wrt } \Delta$ and

$$(p, \varepsilon) \xrightarrow{v''}^{str}_{shp} (q', \varepsilon)$$

wrt Δ . Hence, we can apply the following rule:

$$\frac{q \notin \mathcal{Q}_{shp}^\Delta \quad (q, \varepsilon) \xrightarrow{v'}^{str}_{shp} (p, \varepsilon) \text{ wrt } \Delta \quad (p, \varepsilon) \xrightarrow{v''}^{str}_{shp} (q', \varepsilon) \text{ wrt } \Delta}{(q, \varepsilon) \xrightarrow{v' \cdot v''}^{str}_{shp} (q', \varepsilon) \text{ wrt } \Delta}$$

This shows $(q, \varepsilon) \xrightarrow{v}_{shp}^{str} (q', \varepsilon)$ as required.

Subcases $h = \langle h' \rangle$ or $h = a$ or $h = \varepsilon$. Straightforward.

For the implication from the right to the left we assume that $(q, \varepsilon) \xrightarrow{v}_{shp}^{str} (q', \varepsilon) \text{ wrt } \Delta$.

Case $q \in \mathcal{Q}_{shp}^\Delta$. Then, the following rule must have been applied with $q = q'$:

$$\frac{q \in \mathcal{Q}_{shp}^\Delta \quad v = nw(\mathcal{H}_\Sigma)}{(q, \varepsilon) \xrightarrow{v}_{shp}^{str} (q, \varepsilon) \text{ wrt } \Delta}$$

Since we assume that v does not have any blocking partial runs with Δ , there exists a partial run on v with Δ that starts with q . Since v is a nested word, any partial

run on v is the nested word of some run, so there exists a run on the hedge h that starts with q . This shows the existence of some state p such that $q \xrightarrow{h} p$ wrt Δ . Since $q \in \mathcal{Q}_{shp}^\Delta$, Lemma 6 shows $q = p$, and thus, $q \xrightarrow{h} q$ wrt Δ . Proposition 8 then proves that $(q, \varepsilon) \xrightarrow{nw(h) \text{ str}} (q, \varepsilon)$ wrt Δ , which is equal to $(q, \varepsilon) \xrightarrow{v \text{ str}} (q', \varepsilon)$ wrt Δ , as required.

Case $q \notin \mathcal{Q}_{shp}^\Delta$. We distinguish all possible forms of the hedge h .

Subcase $h = h' \cdot h''$. Let $v' = nw(h')$ and $v'' = nw(h'')$ so $v = v' \cdot v'' = nw(h)$. The following rule must have been applied:

$$\frac{q \notin \mathcal{Q}_{shp}^\Delta \quad (q, \varepsilon) \xrightarrow{v' \text{ str}}_{shp} (p, \sigma) \text{ wrt } \Delta \quad (p, \sigma) \xrightarrow{v'' \text{ str}}_{shp} (q', \varepsilon) \text{ wrt } \Delta}{(q, \varepsilon) \xrightarrow{v' \cdot v'' \text{ str}}_{shp} (q', \varepsilon) \text{ wrt } \Delta}$$

Since v' is a nested word, it follows that $\sigma = \varepsilon$ too. By the induction hypothesis applied to the nested words v' and v'' , we get $(q, \varepsilon) \xrightarrow{v' \text{ str}} (p, \varepsilon)$ wrt Δ and $(p, \varepsilon) \xrightarrow{v'' \text{ str}} (q', \varepsilon)$ wrt Δ . Hence, we can apply the following rule:

$$\frac{(q, \varepsilon) \xrightarrow{v' \text{ str}} (p, \varepsilon) \text{ wrt } \Delta \quad (p, \varepsilon) \xrightarrow{v'' \text{ str}} (q', \varepsilon) \text{ wrt } \Delta}{(q, \varepsilon) \xrightarrow{v' \cdot v'' \text{ str}} (q', \varepsilon) \text{ wrt } \Delta}$$

This shows $(q, \varepsilon) \xrightarrow{v \text{ str}} (q', \varepsilon)$, as required.

Subcases $h = \langle h' \rangle$ or $h = a$ or $h = \varepsilon$. Straightforward.

This ends the proofs of both directions. $\square$

A streaming evaluator with subhedge projection for deterministic $\text{SHA}^\downarrow$ s A on hedges h can thus be obtained by computing the streaming transition relation with subhedge projection for A of $nw(h)$ starting with the initial configuration. This costs time at most $O(1)$ per letter of $nw(h)$, i.e., constant time per event of the stream.

9.3. Streaming Complexity of Earliest Membership with Subhedge Projection

Using streaming evaluators for $\text{SHA}^\downarrow$ s, we can test earliest membership with subhedge projection in streaming mode by running the $\text{SHA}^\downarrow A^{cgr}(\mathbf{S})_{e(\mathbf{S})}$.

Corollary 3. For any complete dSHAs $(\Sigma, \mathcal{Q}, \Delta, I, F)$, $F \subseteq F_S \subseteq \mathcal{Q}$, language $L = \mathcal{L}(A)$, and schema $\mathbf{S} = \mathcal{L}(A[F/F_S])$, earliest membership with complete subhedge projection for L wrt $\mathbf{S}$ can be tested in streaming mode for any hedge $h \in \mathbf{S}$ in time $O(|1|)$ per nonprojected letter of $nw(h)$ after a preprocessing time of $O(|A^{cgr}(\mathbf{S})_{e(\mathbf{S})}| + n^3d + ns)$, where d is the number of difference relations and s the number of safe subsets in states of $\mathcal{Q}_{e(\mathbf{S})}^{cgr}(\mathbf{S})$. The space required is in $O(\text{depth}(h) + |A^{cgr}(\mathbf{S})_{e(\mathbf{S})}|)$.

Proof. By Theorem 4, the $\text{SHA}^\downarrow A^{cgr}(\mathbf{S})_{e(\mathbf{S})}$ is earliest and sound and complete for subhedge projection. By Propositions 8 and 9, we can thus obtain a streaming membership tester with subhedge projection for a hedge $h \in \mathbf{S}$ by running the streaming evaluator with subhedge projection of $A^{cgr}(\mathbf{S})_{e(\mathbf{S})}$ on the nested word $nw(h)$. Since this $\text{SHA}^\downarrow$ is deterministic, the streaming evaluation can be done in time $O(|1|)$ per letter of $nw(h)$ after a preprocessing time of $O(|A^{cgr}(\mathbf{S})_{e(\mathbf{S})}| + n^3d + ns)$. $\square$

As for safe-no-change projection, the exponential preprocessing time can again be avoided by creating the part of the $\text{SHA}^\downarrow A^{cgr}(\mathbf{S})_{e(\mathbf{S})}$ needed for the evaluation of the hedge on the fly. The size of the needed part is in $O(|h|)$. Hence, the space requirements can also

be reduced to $O(|h|)$, which may be too much for large streams $nw(h)$. The needed part of $A^{cgr}(\mathbf{S})_{e(\mathbf{S})}$, however, may be much smaller than $|h|$. Furthermore, a stack of size $depth(h)$ has to be maintained but this dependence on h may be acceptable even for large streams $nw(h)$.

10. Monadic Regular Queries and XPath

We next consider the problem of how to answer monadic regular queries on hedges in an earliest manner and with subhedge projection. Each monadic query defines a set of positions for each input hedge. The XPath queries `self::list[child::item]`, for instance, can be applied to the hedge $\langle \text{list}.\langle \text{list} \rangle.\langle \text{item} \rangle \rangle.\langle \text{list}.\langle \text{item} \rangle \rangle$, where it selects the position of the first top-most tree labeled by `list` since it contains a child tree labeled by `item`. We identify positions of hedges by integers. For instance, the selected `list` element in the hedge above is identified by the integer 2. Therefore, the answer set of the above query on the above hedge is $\{2\}$. Note that monadic queries do not return any other information about the selected positions, in contrast to the subtree semantics of XPath in the official W3C standard.

Answering this query is more complicated than verifying whether the XPath filter `[self::list/child::item]` matches at the root of the first subtree of the hedge, since the answer set needs to be constructed, too. Concerning memory consumption, alive answer candidates need to be buffered for monadic query answering. Therefore, the memory consumption of streaming evaluators of monadic queries may grow linearly in the size of the stream to the number of the alive candidates does.

10.1. Monadic Regular Queries

Monadic queries on hedges in $\mathcal{H}_\Sigma$ can be identified with languages of annotated hedges in $\mathcal{H}_{\Sigma \cup \{x\}}$, where $x \notin \Sigma$ in an arbitrarily fixed selection variable. A monadic query on hedges in $\mathcal{H}_\Sigma$ can then be identified with the language of all annotated hedges in $\mathcal{H}_{\Sigma \cup \{x\}}$, in which a single position got x -annotated and this position is selected by query.

Therefore, regular monadic queries on hedges in $\mathcal{H}_\Sigma$ can be defined by nested regular expressions in $nRegExp_{\Sigma \cup \{x\}}$ or by a dSHA with the same signature. The regular XPath query `self::list[child::item]`, for instance, can be defined by following nested regular expression:

$$\langle \text{list} \cdot x \cdot \top \cdot \langle \cdot \text{item} \cdot \top \cdot \rangle \cdot \top \rangle \cdot \top,$$

where $\top$ is like a wildcard accepting any hedge in $\mathcal{H}_\Sigma$ without x , i.e., for some recursion variable $z \in V$:

$$\top = \mu z. (+_{a \in \Sigma} a + \langle z \rangle)^*.$$

Note that the nested regular expression for the XPath filter `[self::list/child::item]` in $nRegExp_\Sigma$ can be obtained from that of the XPath query `self::list[child::item]` by removing the x . In addition, for any monadic query on hedges, we have to restrict the annotated hedges to be recognized by the language of the query such that exactly one position in each hedge is annotated by x . This can be done by intersection with the following nested regular expression one_x :

$$\mu z. (T.x.T + T.\langle z \rangle.T)$$

The previous schema $\llbracket \text{doc} \rrbracket$ for pattern matching has now to be changed from to $\mathbf{S} = \llbracket \text{doc}_x \rrbracket \cap \llbracket one_x \rrbracket$, where $\text{doc}_x \in nRegExp_{\Sigma \cup \{x\}}$ is like doc but over the signature $\Sigma \cup \{x\}$. For the regular XPath query `self::list[child::item]` we obtain the dSHA in Figure 22. The transition rules with the special letter x indicate, where positions may be bound to the selection variable: x can be bound only in state 2, that is in subtrees starting with a `list` element. The same automaton can be used to define the schema $\mathbf{S} = \llbracket \text{doc}_x \rrbracket \cap \llbracket one_x \rrbracket$ by using the schema-final states $F_S = \{14, 20\}$ instead of the final states $F = \{20\}$. Indeed, we constructed the automaton by intersecting the automaton obtained from the XPath expression with the automata computed from the regular expressions doc_x and one_x for

the schema. Note that 20 is a final state for the query (and the schema), while 14 is a sink state for the query (but not for the schema).

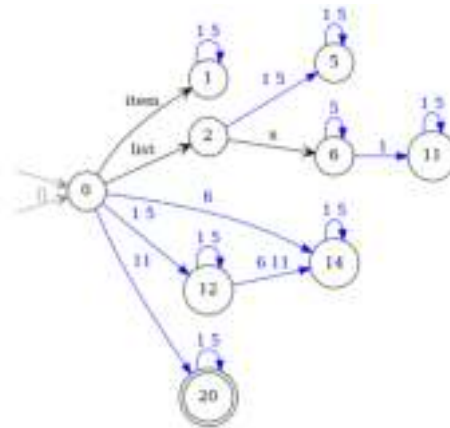


Figure 22. The dSHA for the XPath query `self::list[child::item]`. The selection position is indicated by x .

10.2. Earliest Query Answering with Subhedge Projection

A naive query evaluation algorithm for a hedge automaton with signature $\Sigma \cup \{x\}$ receives an input hedge $h \in \mathcal{H}_\Sigma$, guesses all possible positions of where to insert x into h , and tests for all of them whether the annotated hedge obtained is accepted by A . In a less naive manner, one inserts x only at those position where the current state has an outgoing x transition. All this can be improved if the hedge automaton under consideration supports selection, rejection, or projection states. In all cases, the algorithm has to buffer all binding candidates, that are not yet in a selection or rejection state. In the worst case it has to buffer the bindings to all positions. Therefore, the state space of the evaluator of monadic queries is no more finite.

Selection and rejection states can be identified by adapting the earliest query answering algorithm from [21]. This can be made such that one obtains an $\text{SHA}^\downarrow A_{e(S)}$ with signature $\Sigma \cup \{x\}$ depending on the schema. It should be noticed that automata used in [21] are dNWA's instead of SHA's, and that schemas are ignored there. The needed adaptations are therefore discussed in Appendix A. How to obtain an efficient query answering algorithm in streaming mode from $A_{e(S)}$ is the main topic of [21]. This is the base algorithm that we will extend with subhedge projection for our experiments.

In order to add congruence projection to the earliest query answering algorithm, we have to run the automaton $A_{e(S)}^{gr}(S)$. The earliest congruence projection for the XPath query `self::list[child::item]` is shown in Figure 23. It binds x in state 2D1 after having read a top-level `list`-element and then checks for the existence of a child element `item`, going into selection state 1D3 once it is found. All other children of the top-level `list` element are projected in state 2D3. For adding earliest query answering to self-no-change projection, we have to run the automaton $A_{e(S)}^\pi$ instead of $A_{e(S)}^{gr}$. We do not give the details, but claim for the present query that the projection power of both automaton is the same.

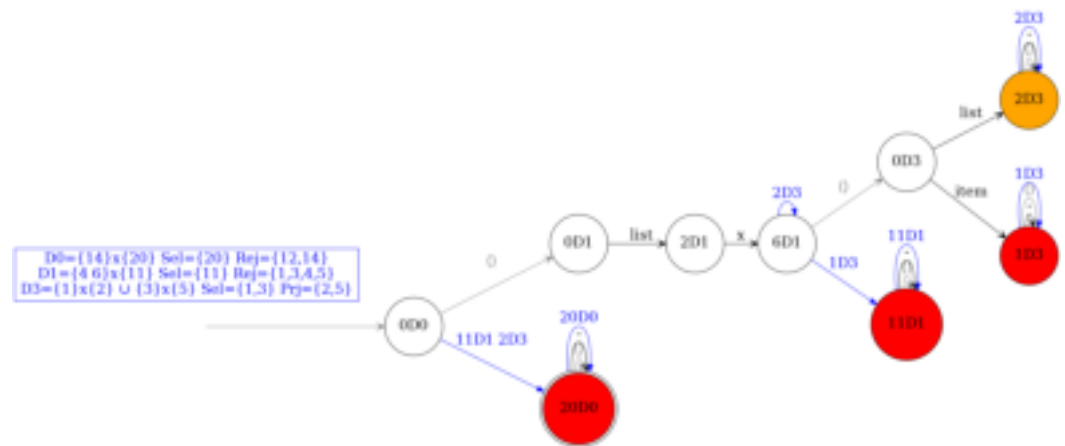


Figure 23. The earliest congruence projection $A^{cgr}(\llbracket doc \rrbracket \cap \llbracket one_x \rrbracket)_{e(\llbracket doc \rrbracket \cap \llbracket one_x \rrbracket)}$ for the dSHA A in Figure 22 with $F_S = \{14, 20\}$. It defines the query of the XPath `self::list[child::item]`.

Fortunately, the soundness and completeness theorems for earliest membership testing with subhedge projection do only affect the constructions of these $SHA^\downarrow$ s, but not how they are to be evaluated. However, there is one additional issue: Nested word prefixes cannot be considered as irrelevant for the subsequent subhedge, if the selection variable x can be bound there, even if the acceptance doesn't depend on where it will be bound. This is since the position of the binding is to be returned by any query answering algorithm.

Definition 15. We call a prefix v binding irrelevant for L and S if there does not exist any hedge h containing x and suffix w such that $v \cdot w \in nw(S)$ and $v \cdot nw(h) \cdot w \in nw(L)$.

Our subhedge projecting evaluator can project only at subhedge irrelevant prefixes that are also binding irrelevant. Given an dSHA A and a set of schema final-states F_S , one can decide whether a prefix is subhedge irrelevant if the current state q does accept hedges containing x but can access some other state that is not safe for rejection and permits x -bindings. In ground Datalog, we can distinguish binding irrelevant states q by imposing the following rules for all $q, q' \in Q$:

```
binding_irrelev(q) :- state(q), not binding_relev(q).
binding_relev(q) :- not bind_x(q), acc(q, q'), bind_x(q'), not rej(q').
```

Note in our example of the XPath query `self::list[child::item]` that state 2D3 of the $dSHA^\downarrow A^{cgr}(\llbracket doc \rrbracket \cap \llbracket one_x \rrbracket)_{e(\llbracket doc \rrbracket \cap \llbracket one_x \rrbracket)}$ in Figure 23 is binding irrelevant, since x must be bound on the top-level list element in state 2D1, so it cannot be bound on any list element below.

11. Experiments with Streaming XPath Evaluation

We compare experimentally three streaming evaluators for dSHA, for earliest monadic query answering, without subhedge projection, with safe-no-change projection, and with congruence projection.

11.1. Benchmark dSHA for XPath Queries

We start from dSHA defining monadic queries for the regular XPath queries A1-A8 from the XPathMark benchmark [7] that are given in Table 1. These XPath queries show most of the features of the regular fragment of XPath. In Table 2, we added 14 further XPath path queries that we found useful for testing too.

Table 1. Regular XPath queries A1-A8 from the XPathMark collection.

Query ID	XPath Expression
A1	/site/closed_auctions/closed_auction/annotation/description/text/keyword
A2	//closed_auction//keyword
A3	/site/closed_auctions/closed_auction//keyword
A4	/site/closed_auctions/closed_auction[annotation/description/text/keyword]/date
A5	/site/closed_auctions/closed_auction[descendant::keyword]/date
A6	/site/people/person[profile/gender and profile/age]/name
A7	/site/people/person[phone or homepage]/name
A8	/site/people/person[address and (phone or homepage) and (creditcard or profile)]/name

Table 2. Additional regular XPath queries for the XPathMark documents.

A0	/site
A1_0a	/site/*
A1_0b	/site/@*
A1_0c	/site//@*
A1_1a	//bidder/personref[starts-with(@person, 'person0')]
A1_1d	//bidder/personref[@person='person0']
A1_2	//person
A1_3	/site/regions/africa/@*
A1_4	/site/regions/africa/*
A1_5	/site/regions/*
A1_6	//closed_auction/annotation//keyword
A2_1	//closed_auction[descendant::keyword]
A4_0	/site/closed_auctions/closed_auction[annotation]/date
A4_1	/site[open_auctions]/closed_auctions

Deterministic SHAs for A1-A8 were provided earlier in [21]. For the other XPath queries we compiled them to dSHAs via nested regular expression.

In order to produce the input dSHAs for our evaluators, we intersect these automata with a dSHA for the schema $\llbracket \text{doc}'_x \rrbracket \cap \llbracket \text{one}_x \rrbracket$ where doc'_x is the schema we use for hedge encodings of real-world XML documents with x annotations. Thereby, we could identify the schema final states F_S . We minimized and completed the result. Table 3 reports the size of the input dSHAs for our evaluators obtained by the above procedure. For each dSHA A , the size is given by two numbers $\text{size}(n)$, the first for the overall size and the second for the number of states n .

For the input dSHA A of each query, we statically computed the whole $\text{dSHA}^\downarrow_s A^{cgr}(\mathbf{S})_{e(\mathbf{S})}$ while using the necessary parts of the determinization algorithm from [16]. The size of the $\text{dSHA}^\downarrow_s$ obtained and the number d of difference relations are reported in Table 3 too. The biggest size is 2091(504) for the input dSHA for A8. The largest number of difference relations $d = 24$ is also obtained for this query. So, indeed the size of these automata is much

smaller than one might expect from a construction that is highly exponential in the worst case. However, the only query that we couldn't obtain its correspondent deterministic earliest congruence projection $A^{cgr}(\mathbf{S})_{e(\mathbf{S})}$ is A5.

The time for computing the earliest congruence projection took between 2.3 and 26 s.

Table 3. Overall size and number n of states of the input dSHA A for the query with schema-final states F_S , the overall size and number of states of the $\text{SHA}^\downarrow A^{cgr}(\mathbf{S})_{e(\mathbf{S})}$ obtained by earliest congruence projection, and the number d its difference relations.

Query ID	dSHA A Size(n)	dSHA $^\downarrow$ Size(#State)	#Diff-Rel. d	Query ID	dSHA A Size(n)	dSHA $^\downarrow$ Size(#State)	#Diff-Rel. d
A1	482(68)	1296(324)	16	A1_0c	230(43)	238(62)	6
A2	224(42)	316(82)	7	A1_1a	305(54)	384(101)	8
A3	320(53)	662(156)	10	A1_1d	305(54)	382(101)	8
A4	629(74)	1651(404)	18	A1_2	194(39)	152(42)	5
A5	438(63)	1226(269)	13	A1_3	318(53)	672(159)	10
A6	675(76)	2090(500)	22	A1_4	312(52)	479(132)	10
A7	394(59)	728(184)	12	A1_5	266(47)	293(84)	8
A8	648(79)	2091(504)	24	A1_6	265(47)	588(142)	9
A0	203(40)	158(44)	5	A2_1	232(42)	295(78)	7
A1_0a	224(42)	145(44)	6	A4_0	392(59)	719(184)	12
A1_0b	203(39)	68(23)	5	A4_1	292(49)	287(78)	8

We note that we have not yet computed the complete dSHA $^\downarrow$ s statically for safe-no-change projection A^{snc} , pure congruence projection $A^{cgr}(\mathbf{S})$ and earliest query answering $A_{e(\mathbf{S})}$. We believe that these automata may become bigger than $A^{cgr}(\mathbf{S})_{e(\mathbf{S})}$ that we constructed in a single shot.

11.2. Streaming Evaluation Tool: AStream

We are using and developing the AStream tool for answering monadic dSHA queries on XML streams with schema restrictions. Version 1.01 of AStream was presented in [21] supports earliest query answering without projection. Given a dSHA A defining a monadic query, a set of schema final states F_S and an XML stream w , it constructs on-the-fly the needed part of the $\text{SHA}^\downarrow A_{e(\mathbf{S})}$ while parsing w and evaluating A on the hedge encoding w .

For the FCT conference version of the present paper [24], we enhanced AStream with safe-no-change projection. This leads to version AStream 2.01. It constructs the earliest safe-no-change projection dSHA $^\downarrow A^{snc}_{e(\mathbf{S})}$ on the fly while evaluating the monadic query defined by A on the hedge encoding the XML stream w . Subhedge projection can be switched on or off in order to compare both versions without further implementation differences.

For this present journal version, we added earliest congruence projection and integrated it into AStream leading to AStream 3.0.

AStream 3.0 is different from the two previous versions in that the dSHA $^\downarrow A^{cgr}_{e(\mathbf{S})}$ is constructed statically and entirely, independently of the input hedge. We then use a generic earliest streaming evaluator for $\text{SHA}^\downarrow$ s that rejects in rejection states, selects in selection states, and projects in all subhedge projection states. This evaluator could also be run with $A^{snc}_{e(\mathbf{S})}$ or $A_{e(\mathbf{S})}$, as long as these do not grow too big.

It should be noticed that $A^{cgr}_{e(\mathbf{S})}$ turned out to be nicely small for our whole benchmark, with the other dSHA $^\downarrow$ s $A^{snc}_{e(\mathbf{S})}$ and $A_{e(\mathbf{S})}$ risk becoming bigger. As stated earlier, we did not construct these dSHA $^\downarrow$ s so far for our benchmark queries. This is why we continued to run

the earliest streaming with safe-no-change projection with AStream 2.01, while we used AStream 3.0 for earliest streaming with congruence projection.

All AStream versions rely on Java's abc-Datalog for computing the least fixed points of Datalog programs in a bottom-up manner. Datalog programs are needed for all logical reasoning: for computing subsets of states that are safe for rejection, selection, or subhedge projection on all levels. Also, the difference relations are computed based on Datalog. Note that earliest on-the-fly query evaluation requires running Datalog during query evaluation, while with a static approach for constructing $dSHA^\downarrow$ s entirely, Datalog is only needed at preprocessing time during the automaton construction.

11.3. Evaluation Measures

We want to measure the time for running the three streaming query evaluators for all dSHAs obtained from the XPath expressions in the collection in Tables 1 and 2.

One advantage of the XPathMark benchmark is that it comes with a generator of XML documents to which the queries can be applied and that can be scaled in size. We created an XML document of size 1.1 GB, which is sufficiently large to make streaming mandatory. Our experiments show that the efficiency of query evaluation grows linearly with the size of the non-projected part of the XML document. This holds for each of our three evaluators. Therefore, measuring the time of the evaluator on XML documents of other sizes would not show any new insights, except that memory consumption remains moderate too.

The time gain of projection for a query Q is the percentage of time saved by projection during query evaluation when ignoring the pure document parsing time t_{parse} seconds:

$$\text{time-gain}_Q = 100 * (1 - (t_{\text{noproject}_Q} - t_{\text{parse}}) / (t_{\text{project}_Q} - t_{\text{parse}})).$$

We can measure the time gain for earliest safe-no-change projection by AStream 2.01 and for earliest congruence projection by AStream 3.0, since projection can be switched off in both of them. The disadvantage of the time gain is that it depends on the details of the implementation.

The event gain is a better measure for the projection power. The parser sends a stream of needed events to the $SHA^\downarrow$, while ignoring parts of the input document that can be projected away. For this, it must always be informed by the automaton about what kind of events can be projected in the current state. The event gain of projection event-gain_Q is then the percentage of events that are recognized to be irrelevant and thus no more created by a projecting evaluator for query Q . One might expect that the time gain is always smaller than the event gain.

$$\text{time-gain}_Q \leq \text{event-gain}_Q$$

Indeed, this will be confirmed by our experiments. We believe that these discrepancies between these two measure indicate how much room remains for optimizing the implementation of an evaluator. In this way, we can observe that some room for further optimizations AStream 3.0 still remains.

11.4. Experiments

We used Java's *XMLStreamReader* Interface v1.0 from the *javax.xml.stream* package to parse and stream XML files in Scala. Parsing a 1.1 GB XML document required $t_{\text{parse}} = 15$ s, while querying it without projection took us in average $t_{\text{noproject}_{\text{avg}}} = 752$ s, while varying from 600 to 900 s. The average processing time for 1 MB was thus 0.72 s. Once we knew this, we predicted the expected evaluation time from the size of the non-projected part of the XML document. It is as follows:

$$\text{event-gain}_Q * 1100 * 0.72 \text{ s} + t_{\text{parse}} \text{ s}.$$

Without projection, the parser generates 1,012,018,728 events for the 1.1 GB XML document. This is the baseline for computing event-gain_Q for all queries Q .

We ran AStream with Scala v2.13.3 on a machine with the operating system Ubuntu 20.04.06 LTS (64-bit). The machine is a Dell Precision-7750 machine, equipped with an Intel® Core™ i7-10875H CPU @ 2.30GHz × 16 and 32 GB of RAM.

11.4.1. Earliest Congruence Projection

We present the measures of our evaluator with the earliest congruence projection in Table 4. The event gain is high, varying between 75.7% for A1_0c and 100% for A0, A1_0a, A1_0b, A1_4, A1_5, and A4_1. The event gain for queries without descendant axis is above 98.2%. For these queries, all subtrees that are not on the main path of the query can be projected away. In particular, only subtrees until a fixed depth have to be inspected. These are the queries A1, A4, A6–A8 and A4_0 and the queries with event gain 100% listed above. The time gain for all these queries is a little smaller than the event gain but still above 98.2%.

Table 4. Time gain and event gain by earliest congruence projection with AStream3.01 on a 1.1 GB XPathMark stream.

Query ID	#Ans-Wer	Time in s	Time Gain	Event Gain	Query ID	#Ans-Wer	Time in s	Time Gain	Event Gain
A1	38,267	29.3	98.1%	98.9%	A1_0c	3,600,816	226.6	72.9%	75.7%
A2	117,389	171.8	77.9%	81.1%	A1_1a	2	187.8	78.1%	80.3%
A3	117,389	33.4	97.4%	97.8%	A1_1d	2	217.3	74.8%	80.3%
A4	24,959	33.2	97.8%	98.9%	A1_2	1,062,965	238.2	68.2%	76.0%
A5	50,186				A1_3	25074	16.2	99.1%	99.8%
A6	30,307	45.9	96.9%	98.2%	A1_4	5170	17.5	99.7%	100.0%
A7	179,889	30.5	98.0%	98.7%	A1_5	6	14.3	100.0%	100.0%
A8	67,688	37.8	97.3%	98.7%	A1_6	117,389	188.7	75.4%	81.1%
A0	1	21.3	99.1%	100.0%	A2_1	50,186	199.7	76.5%	81.1%
A1_0a	6	15.2	99.9%	100.0%	A4_0	91,650	21.8	99.0%	99.3%
A1_0b	0	0.0	100.0%	100.0%	A4_1	1	12.0	100.0%	100.0%

We next consider the second type of queries having the descendant axis, specifically A2, A1_0c, A1_1a, A1_1d, A1_2, A1_6, and A2_1. Intuitively, a lower event gain is expected in these cases because subhedge projection with the descendant axis cannot directly exclude an entire subtree under a given element; all descendant elements must be inspected. For instance, the query A1_0c equal to `/site//@*` must select all attributes of all elements under the root element *site*, requiring the inspection of every XML element in the document, for attribute presence, except for the root element *site*.

The lower event gain reported for the aforementioned queries align with the above expectation, showing percentages ranging from 75.7% to 81.1%. Also, these queries exhibit a larger gap between time gain and event gain, which ranges from 2.2% to 5.7% and will be subject to future optimization. It is worth noting that some of these queries combine multiple features at once, like queries A1_1a and A1_1d, which have a descendant axis, filter with attributes, and string comparison.

The only exception to this trend for the queries having descendant axis are A3 and A1_3, with respective event gains of 97.8% and 99.8%. The reason behind this high gain is that these queries starts from the root, with a child axis path of depth 3 before querying the descendant axis. This initial path allows for the exclusion and projection of most elements under *site* that do not fit the specified path with their entire subtrees.

11.4.2. Earliest Safe-No-Change Projection

We compare earliest safe-no-change projection with the earliest congruence projection in Table 5. For this, we restrict ourselves to the queries A1–A8 from the XPathMark in Table 1.

The XPath queries A1, A4, A6, A7, and A8 do not have descendant axes. The time gain for safe-no-change projection on these queries is between 92.3–98.9%. For earliest congruence projection, the time gain was better for A1, A4, and A6 with at least 96.6%. For A7 and A7, the time gain of safe-no-change projection is close to the event gain of congruence projection but the time gain of congruence projection is 1.5% lower. So, we hope that an optimized version of congruence projection could outperform safe-no-change projection in all cases.

The XPath queries with descendant axis are A2, A3, and A5. For A2 and A3 in particular, the time gain of safe-no-change projection is very low with even not 10.9% while congruence projection yields at least 77.9%. We believe that this is bug in our current implementation of safe-no-change projection, which may be prohibiting the projection of attributes and text nodes for these queries (but not for the others). Congruence projection on A2 gains 77.9% of time and 81.1% of events. This is much better, but still far from the projection power for queries without descendant axis. For A3, congruence projection gains 97.4% of time and 97.8% of events. This is much better than for A2 since the descendant axis in A3 is at the end, while for A2 it is at the beginning of the path. But, even for A2, congruence projection is very helpful.

Table 5. Time gain for earliest safe-no-change and earliest congruence projection, and the event gain of earliest congruence projection.

Query ID	Time (s) Snc	Time (s) Congr	Time Gain Snc	Time Gain Congr	Event Gain Congr
A1	72.8	29.3	92.3%	98.1%	98.9%
A2	664.6	171.8	8.5%	77.9%	81.1%
A3	666.1	33.4	10.9%	97.4%	97.8%
A4	78.5	33.2	92.4%	97.8%	98.9%
A5	77.7		92.3%		
A6	65.2	45.9	95.1%	96.9%	98.2%
A7	24.6	30.5	98.8%	98.0%	98.7%
A8	24.8	37.5	98.9%	97.3%	98.7%

11.4.3. Comparison to External Tools

QuiXPath was shown to be the most efficient large coverage external tool for regular XPath evaluation on XML streams [4]. Therefore, we compare the efficiency of congruence projection to QuiXPath in Table 6, and thereby, by transitivity to the many alternative tools. We note that QuiXPath performs early query answering by compilation to possibly nondeterministic NWA with selection states, without always being earliest. Apart from this, it bases its efficiency on both subtree projection and descendant projection.

Table 6. Comparison of of parse-free timings in seconds for XPathMark queries with QuiXPath and earliest congruence projection.

Query ID	Parse-Free Time (s) QuiXPath	Parse-Free Time (s) Congr. Prj.	Fract.	Query ID	Parse-Free Time (s) QuiXPath	Parse-Free Time (s) Congr. Prj.	Fract.
A1	11.0	14.3	1.3	A2	11.4	156.8	13.8
A4	11.6	18.2	1.6	A3	11.5	18.4	1.6
A6	10.7	30.9	2.9	A5	12.0		
A7	8.6	15.5	1.8				
A8	8.8	22.5	2.6				

The experiments with QuiXPath in [4] use the parse-free time. Therefore, we do the same for congruence projection by reducing the measured overall time by 15 s of parsing time.

Compared to the parsing-free times for the XPath queries without descendant axis, the earliest congruence projection demonstrates significant improvements. On average, our projection is only slower by a factor of 1.3–2.9 than QuiXPath. This shows that our implementation is already close to be competitive with the existing XML streaming tools, while being the only one guaranteeing earliest query answering.

For query A2 with a descendant axis, congruence projection is by a factor of 13.8 slower than QuiXPath. The reason is that QuiXPath supports descendant projection, while congruence projection concerns only subhedge. For query A3 with a descendant axis at the end, congruence projection is only by a factor of 1.6 behind, so descendant projection seems less relevant here.

12. Related Automata Models

A large number of alternative automata notions for trees, forests, and hedges were proposed in the literature. We compare them to SHAs and SHA⁺s in the following subsection and discuss the implications.

12.1. SHAs versus Tree, Forest, and Hedge Automata

Standard Hedge Automata. Standard hedge automata [9,10,32,33] operate on labeled hedges in a bottom-up manner, similarly to SHAs. Also they have the same expressiveness, but horizontal languages are specified differently, leading to a problematic notion of determinism [34]. For this reason, unique minimization fails for deterministic standard hedge automata. Still, they have the same expressiveness as SHAs when restricted to labeled hedges.

Standard Tree Automata and SHA Minimization. Syntactically, any SHA A is a standard tree automaton over the following ranked signature:

- a unary symbol for any letter $a \in \Sigma$;
- a binary symbol @;
- a constant $\langle \rangle$.

Semantically, there is no perfect general correspondence. Only for subclasses a perfect correspondence can be made up via binary encoding. This was shown in [8] for the subclass of dSHAs with $I = \langle \rangle^\Delta$. In [35], it could also be shown for the subclass of multi-module dSHAs. This leads to unique minimization results for both subclasses of deterministic SHAs.

Bojanczyk's Forest Automata. Standard hedge automata also have the same expressiveness as Bojanczyk's forest automata (see Section 3.3 of [36]). These are based on the idea of transition monoids, rather than on the idea of transition rules such as SHAs or finite

state automata or Neuman's and Seidl's forest automata. As a consequence, there is an exponential difference in succinctness between SHAs and Bojanczyk's forest automata [35].

12.2. $\text{SHA}^\downarrow$ s versus SHAs

We start explaining why $\text{SHA}^\downarrow$'s have the same expressiveness as SHAs. Basically, it is the same reason for why NWA's have the same expressiveness as SHAs (see e.g., [8]). For the easier direction, we argued already that any SHA $A = (\Sigma, \mathcal{Q}, \Delta, I, F)$ can be converted to a $\text{SHA}^\downarrow \text{down}(A)$.

Downward Elimination. Conversely, we can convert any $\text{dSHA}^\downarrow A$ into an equivalent SHA $\text{elim}^\downarrow(A)$ while possibly introducing nondeterminism as follows:

$$\begin{array}{c} I^{\text{elim}^\downarrow} = I \times \mathcal{Q} \\ F^{\text{elim}^\downarrow} = F \times \mathcal{Q} \end{array} \quad \frac{q \xrightarrow{\langle \rangle} q' \text{ in } \Delta}{\xrightarrow{\langle \rangle} (q, q') \text{ in } \Delta^{\text{elim}^\downarrow}} \quad \frac{q \xrightarrow{a} q' \text{ in } \Delta \quad r \in \mathcal{Q}}{(r, q) \xrightarrow{a} (r, q') \text{ in } \Delta^{\text{elim}^\downarrow}} \quad \frac{q @ p \rightarrow q' \text{ in } \Delta \quad r \in \mathcal{Q}}{(r, q) @ (q, p) \rightarrow (r, q') \text{ in } \Delta^{\text{elim}^\downarrow}}$$

Proposition 10. $\mathcal{L}(A) = \mathcal{L}(\text{elim}^\downarrow(A))$.

The construction is analogous to the conversion of NWA's to SHAs [8] or to hedge automata [26]. The correctness proofs for these compilers are standard.

Unique Minimization. Unique minimization fails for $\text{dSHA}^\downarrow$ s as usual for deterministic multiway automata. This even happens for deterministic two-way finite state automata on words. In contrast, the subclass of dSHAs with the restriction that the initial state and the tree initial state are the same enjoys unique minimization [8].

Neuman and Seidl's Pushdown Forest Automata. Standard hedge automata are also known to have the same expressiveness than Neumann and Seidl's pushdown forest automata [25]. The latter can be identified with $\text{SHA}^\downarrow$ s for labeled hedges. What is needed to map pushdown forest automata to standard hedge automata is downward elimination, as for mapping $\text{SHA}^\downarrow$ s to SHAs.

12.3. $\text{SHA}^\downarrow$ s versus NWA's

The streaming evaluator for $\text{SHA}^\downarrow$ s via a visibly pushdown machine can also be obtained by compiling a $\text{SHA}^\downarrow$ to a nested word automaton (NWA) [11] whose streaming evaluator is given by a visibly pushdown machine [12], previously known as *input driven automata* [13–15].

Definition 16. A nested word automata (NWA) is a tuple $(\Sigma, \mathcal{Q}, \Gamma, \Delta, I, F)$, where Σ , Γ , and $\mathcal{Q}$ are sets, $I, F \subseteq \mathcal{Q}$, and $\Delta = ((a^\Delta)_{a \in \Sigma}, \langle^\Delta, \rangle^\Delta)$ contains the following relations: $a^\Delta \subseteq \mathcal{Q} \times \mathcal{Q}$, $\langle^\Delta \subseteq \mathcal{Q} \times (\Gamma \times \mathcal{Q})$, and $\rangle^\Delta \subseteq \mathcal{Q} \times \Gamma \times \mathcal{Q}$. A NWA is deterministic or equivalently a dNWA if I contains at most one element and all above relations are partial functions.

The elements of Γ are called stack symbols. The transition rules in Δ again have three forms: internal rules $q \xrightarrow{a} q'$ in Δ as for SHAs, opening rules $q \xrightarrow{\langle^\Delta \gamma} q'$ in Δ if $\langle^\Delta(q) = (q', \gamma)$, and closing rules $q \xrightarrow{\gamma \rangle^\Delta} q'$ in Δ if $\rangle^\Delta(q, \gamma) = q'$.

Streaming evaluators for NWA's. The streaming evaluator for NWA's can be seen as pushdown machines for evaluating the nested words of hedges in streaming manner. A configuration of the pushdown machine is a pair in $\mathcal{K} = \mathcal{Q} \times \Gamma^*$ containing a state and a stack.

For any word $v \in \hat{\Sigma}^*$, we define a streaming transition relation $\xrightarrow{v}^{str} \subseteq \mathcal{K} \times \mathcal{K}$, such that, for all $q, q' \in \mathcal{Q}$ and $\sigma \in \Gamma^*$, the following is applicable:

$$\frac{true}{(q, \sigma) \xrightarrow{\varepsilon}^{str} (q, \sigma) \text{ wrt } \Delta} \quad \frac{(q, \sigma) \xrightarrow{v}^{str} (q', \sigma) \quad (q', \sigma) \xrightarrow{v'}^{str} (q'', \sigma) \text{ wrt } \Delta}{(q, \sigma) \xrightarrow{v \cdot v'}^{str} (q'', \sigma) \text{ wrt } \Delta}$$

$$\frac{q \xrightarrow{a} q' \text{ in } \Delta}{q \xrightarrow{a}^{str} q' \text{ wrt } \Delta} \quad \frac{q \xrightarrow{\downarrow \gamma} q' \text{ in } \Delta}{(q, \sigma) \xrightarrow{\downarrow \gamma}^{str} (q', \sigma) \text{ wrt } \Delta} \quad \frac{q \xrightarrow{\uparrow \gamma} q' \text{ in } \Delta}{(q, \sigma \cdot \gamma) \xrightarrow{\uparrow \gamma}^{str} (q', \sigma) \text{ wrt } \Delta}$$

From $\text{SHA}^\downarrow$ s to NWA. For any $\text{SHA}^\downarrow A = (\Sigma, \mathcal{Q}, \Delta, I, F)$, we can define the NWA $A^{nwa} = (\Sigma, \mathcal{Q}, \Gamma, \Delta^{nwa}, I^{nwa}, F^{nwa})$ while preserving determinism, such that $\Gamma = \mathcal{Q}$, while Δ^{nwa} contains for all $a \in \Sigma$ and $q, p \in \mathcal{Q}$ transition rules:

$$\frac{q \xrightarrow{a} q' \text{ in } \Delta}{q \xrightarrow{a} q' \text{ in } \Delta^{nwa}} \quad \frac{q \xrightarrow{\langle \rangle} q' \text{ in } \Delta}{q \xrightarrow{\langle \downarrow q \rangle} q' \text{ in } \Delta^{nwa}} \quad \frac{q @ p \rightarrow q' \text{ in } \Delta}{p \xrightarrow{\uparrow q} q' \text{ in } \Delta^{nwa}}$$

For instance, the dNWA A^{nwa} of the $\text{dSHA}^\downarrow A$ in Figure 9 obtained from the dSHA for the `[self::list/child::item]` from the introduction is given in Figure 24. The dNWA A^{nwa} of the $\text{dSHA}^\downarrow A$ in Figure 21 is given in Figure 25.

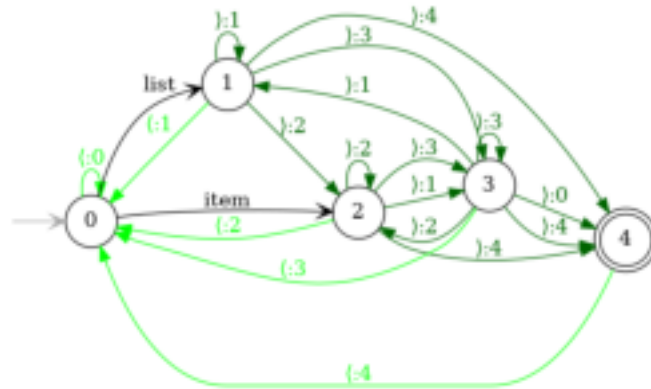


Figure 24. The dNWA A^{nwa} for the $\text{dSHA}^\downarrow A$ in Figure 9 obtained by applying the *down* operator to the dSHA in Figure 4 for the filter `[self::list/child::item]`.

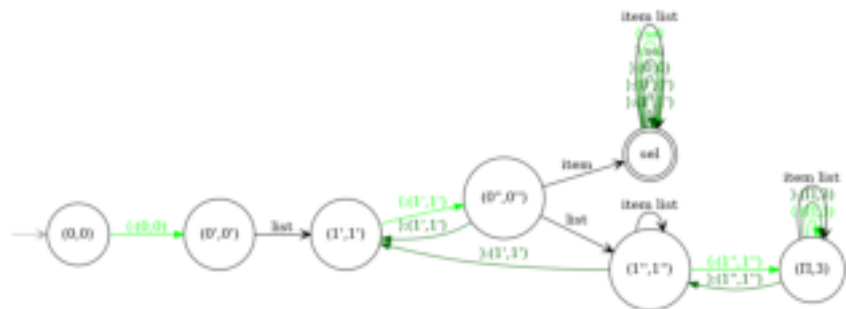


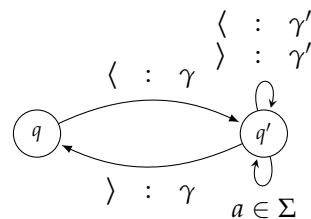
Figure 25. The dNWA $(A_{e([doc])}^{snc})^{nwa}$ for the earliest $\text{dSHA}^\downarrow$ with safe-no-change projection in Figure 21.

Lemma 21. $\mathcal{L}(A^{nwa}) = \mathcal{L}(A)$.

The runs of $\text{SHA}^\downarrow$'s A and NWAs A^{nwa} can be identified.

Projection for NWAs. Projecting evaluators for NWAs were proposed in the context of projecting NWAs [4]. They are based on the following notion of states of irrelevant subtrees for NWAs.

Definition 17 (Variant of Definition 3 of [4]). *We call a state q of an NWA a state of irrelevant subtrees if there exist two different stack symbols γ, γ' and a state q' such that the transitions shown on the right exist but no further opening transitions with γ , no further transitions with γ' , and no further closing transition in q popping γ . In this case, we write $q \in i\text{-tree}_{\Sigma \setminus \emptyset}$.*



Lemma 22. *Any subhedge projection state of a complete $\text{SHA}^\downarrow$ A is a state of irrelevant subtrees of A^{nwa} .*

It was then shown in [4] how to map an NWA with a subset $\mathcal{Q}_{shp}^A$ of states of irrelevant subtrees to a projecting NWA, whose streaming semantics yield a streaming evaluator with subhedge projection. The presentation of subhedge projection for $\text{SHA}^\downarrow$'s without passing via NWAs yields the same result in a more direct manner.

It should also be noticed that projecting NWAs supports descendant projection besides subhedge projection. For SHAs, this is left to future research.

13. Conclusions and Future Work

We introduced the notion of irrelevant subhedges for membership to hedge language under schema restrictions. We developed algorithms with subhedge projection that jump over irrelevant subhedges for testing membership to the regular hedge languages defined by dSHAs. All our algorithms can be run in memory mode and in streaming mode. The difficulty was how to push the needed finite state information for subtree projection top-down given that SHAs operate bottom-up. We solved it based on compilers from SHAs to downward SHAs.

This first compiler propagates safety information regarding non-changing states. The second compiler, which we prove to be complete for subhedge projection, propagates difference relations. We then combined subhedge projection with earliest membership testing and combined it earliest monadic query answering. We integrated both safe-no-change and complete congruence in our AStream tool and confirmed their usefulness experimentally: we showed that it can indeed speed up the the AStream tool for answering regular XPath queries on XML streams. Moreover, congruence projection showed much higher event projection gain rates than safe-no-change, ranging between 75.7% and 100% for the class of queries that does not have descendant axis, becoming competitive in time with the best existing streaming tools for regular XPath queries without a recursive axis.

A remaining open theoretical and practical question is how to obtain descendant projection for SHAs, as needed to speed up the evaluation of XPath queries with a descendant axis. On the implementation and experimental side, it would be nice to investigate the in-memory version of our subhedge algorithms too. Finally, it would be nice to optimize

streaming earliest-query answering tools further until they outperform the best streaming tools in all cases, while being better for queries where the other tools are not earliest.

Author Contributions: Writing—original draft, A.A.S. and J.N. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Dataset available on request from the authors.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A. Earliest Membership

We adapt the earliest membership tester for dSHAS from [21] so that it compiles to $\text{SHA}^\downarrow$ s instead of NWAs and such that it accounts for schemas.

Appendix A.1. Schema Safety Is an Inclusion Problem

The idea for testing certain membership and certain nonmembership is to consider states are safe to reach final states except if going out of the schema. Let $A = (\Sigma, Q, \Delta, I, F)$ be an SHA with schema $\mathbf{S}$, such that there exists $F \subseteq F_{\mathbf{S}} \subseteq Q$ with $S = \mathcal{L}(A[F/F_{\mathbf{S}}])$. Note that $\mathcal{L}(A) \subseteq S \subseteq \mathcal{H}_{\Sigma}$ in particular. The set of states that are safe to reach a subset $P \subseteq Q$ for all hedges that do not go out of the schema $\mathbf{S}$ is as follows:

$$\text{safe}_{\mathbf{S}}^{\Delta}(P) = \text{safe}^{\Delta}(P \cup (Q \setminus F_{\mathbf{S}})).$$

If A is deterministic and schema-complete for $\mathbf{S}$ then the safety of a state of A with respect to schema $\mathbf{S}$ is an inclusion problem.

Lemma A1. *Let $S = \mathcal{L}(A[F/F_{\mathbf{S}}])$. If A is deterministic and schema-complete for $\mathbf{S}$, then for all states $q \in Q$:*

$$\mathcal{L}(A[I/\{q\}, F/F_{\mathbf{S}}]) \subseteq \mathcal{L}(A[I/\{q\}]) \Leftrightarrow q \in \text{safe}_{\mathbf{S}}^{\Delta}(F).$$

Proof. “ $\Leftarrow$ ”. Suppose that $q \notin \text{safe}_{\mathbf{S}}^{\Delta}(F)$. The definition of schema-based safety yields $\text{acc}^{\Delta}(\{q\}) \cap (F_{\mathbf{S}} \setminus F) \neq \emptyset$. So, there exists some hedge $h \in \mathcal{H}_{\Sigma}$, such that $q \xrightarrow{h} F_{\mathbf{S}} \setminus F$ wrt Δ . In particular, $q \in \mathcal{L}(A[I/\{q\}, F/F_{\mathbf{S}}])$. By determinism, there exists no other transition on h from q , and thus, $q \notin \mathcal{L}(A[I/\{q\}])$. Hence, $\mathcal{L}(A[I/\{q\}, F/F_{\mathbf{S}}]) \not\subseteq \mathcal{L}(A[I/\{q\}])$.

“ $\Rightarrow$ ”. Suppose that $q \in \text{safe}_{\mathbf{S}}^{\Delta}(F)$. Let $h \in \mathcal{L}(A[I/\{q\}, F/F_{\mathbf{S}}])$. Then, there exists $q' \in F_{\mathbf{S}}$, such that $q \xrightarrow{h} q'$ wrt Δ . Hence, $q' \in \text{acc}^{\Delta}(\{q\})$ so that $\text{acc}^{\Delta}(\{q\}) \subseteq F \cup (Q \setminus F_{\mathbf{S}})$ implies $q' \in F$. Thus, $h \in \mathcal{L}(A[I/\{q\}])$.

□

Lemma A1 with $F_{\mathbf{S}} = Q$ shows Lemma 5 of [21], which states for any complete dSHAS A that safety is a universality problem:

$$\mathcal{L}(A[I/\{q\}]) = \mathcal{H}_{\Sigma} \Leftrightarrow q \in \text{safe}^{\Delta}(F).$$

This is since the schema-completeness of A for $\mathbf{S} = \mathcal{H}_{\Sigma}$ implies the completeness of A , and thus, $\mathcal{L}(A[I/\{q\}, F/Q]) = \mathcal{H}_{\Sigma}$.

Appendix A.2. Algorithm

We now present our earliest membership tester for dSHAS with schema restrictions. Given a SHA A , we compute a $\text{SHA}^\downarrow A_{e(\mathbf{S})} = (\Sigma, Q_e, \Delta_e, I_e(\mathbf{S}), F_e(\mathbf{S}))$ as follows. Let sel be a

fresh symbol. We start with set set of states that are safe for selection $S_0 = \text{safe}_S^\Delta(F)$ and, respectively, safe for rejection $R_0 = \text{safe}_S^\Delta(\mathcal{Q} \setminus F)$. The state sets of $A_e(s)$ are then as follows:

$$\begin{aligned} \mathcal{Q}_e &= (\mathcal{Q} \times 2^{\mathcal{Q}} \times 2^{\mathcal{Q}}) \cup \{\text{sel}\} \\ I_e(s) &= \{(q, R_0, S_0) \mid q \in I, q \notin R_0 \cup S_0\} \cup \{\text{sel} \mid I \cap S_0 \neq \emptyset\} \\ F_e(s) &= \{(q, R_0, S_0) \mid q \in F\} \cup \{\text{sel}\} \end{aligned}$$

For illustration, reconsider the dSHA from Figure 4. This dSHA is not complete but schema-complete for the schema $\llbracket \text{doc} \rrbracket$. We get $s\text{-down}^\Delta(0, \{4\}) = \{3, 4\}$. It may seem counter intuitive that not only state 3 but also state 4 down remains safe for selection. This reflects the fact that no proper subhedge of any hedge satisfying the intended schema may ever get into state 4. Applying the earliest construction with safe-no-change projection to dSHA in Figure 4 yields the $\text{dSHA}^\downarrow$ in Figure A1. The transition rules in Δ_e are given in Figure A2.

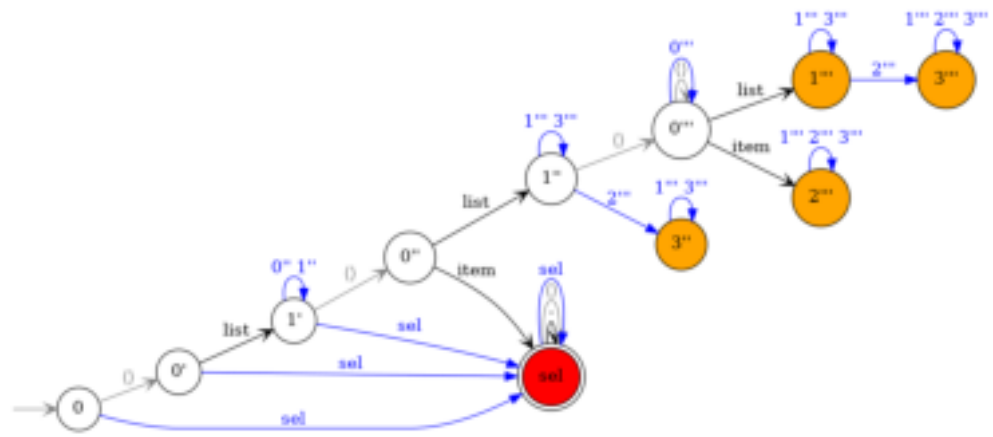


Figure A1. The earliest $\text{dSHA}^\downarrow A_e$ for the dSHA A in Figure 19 with schema $\llbracket \text{doc} \rrbracket$ for the XPath filter $[\text{self}::\text{list}/\text{child}::\text{item}]$. The states of A_e correspond to the following tuples: $0 = (0, \{1, 2, 3, 5\}, \{4\})$, $0' = (0, \{2, 4, 5\}, \{3, 4\})$, $1' = (1, \{2, 4, 5\}, \{3, 4\})$, $0'' = (0, \emptyset, \{2, 4, 5\})$, $1'' = (1, \emptyset, \{2, 4, 5\})$, $0''' = (0, \emptyset, \emptyset)$, $1''' = (1, \emptyset, \emptyset)$, $2''' = (1, \emptyset, \emptyset)$, and $3''' = (1, \emptyset, \emptyset)$.

$$\begin{array}{c}
\frac{q \xrightarrow{a} q' \text{ in } \Delta \quad q' \notin S \cup R}{(q, S, R) \xrightarrow{a} (q', S, R) \text{ in } \Delta_e} \\
\\
\frac{\xrightarrow{\langle \rangle} q' \text{ in } \Delta \quad (q, S, R) \in \text{in } \mathcal{Q}_e \quad q' \notin s\text{-down}^\Delta(q, S) \cup s\text{-down}^\Delta(q, R)}{(q, S, R) \xrightarrow{\langle \rangle} (q', s\text{-down}^\Delta(q, S), s\text{-down}^\Delta(q, R)) \text{ in } \Delta_e} \\
\\
\frac{q @ p \rightarrow q' \text{ in } \Delta \quad q' \notin S \cup R}{(q, S, R) @ (p, s\text{-down}^\Delta(q, S), s\text{-down}^\Delta(q, R)) \rightarrow (q', S, R) \text{ in } \Delta_e} \\
\\
\frac{q \xrightarrow{a} q' \text{ in } \Delta \quad q' \in S}{(q, S, R) \xrightarrow{a} \text{sel} \text{ in } \Delta_e} \quad \frac{\xrightarrow{\langle \rangle} q' \text{ in } \Delta \quad (q, S, R) \in \text{in } \mathcal{Q}_e \quad q' \in s\text{-down}^\Delta(q, S)}{(q, S, R) \xrightarrow{\langle \rangle} \text{sel} \text{ in } \Delta_e} \\
\\
\frac{q @ p \rightarrow q' \text{ in } \Delta \quad q' \in S}{(q, S, R) @ (p, s\text{-down}^\Delta(q, S), s\text{-down}^\Delta(q, R)) \rightarrow \text{sel} \text{ in } \Delta_e} \\
\\
\frac{a \in \Sigma}{\text{sel} \xrightarrow{a} \text{sel} \text{ in } \Delta_e} \quad \frac{\mu \in \mathcal{Q}_e}{\text{sel} @ \mu \rightarrow \text{sel} \text{ in } \Delta_e} \quad \frac{\mu \in \mathcal{Q}_e}{\mu @ \text{sel} \rightarrow \text{sel} \text{ in } \Delta_e} \quad \frac{\text{true}}{\text{sel} \xrightarrow{\langle \rangle} \text{sel} \text{ in } \Delta_e}
\end{array}$$

Figure A2. The earliest transition rules Δ_e inferred from transition rules Δ of some SHA.

Appendix A.3. Soundness and Completeness

We next provide soundness and completeness results for our earliest in-memory membership tester with projection.

Proposition A1 (Soundness). *For any SHA A with schema $S = \mathcal{L}(A[F/F_S])$, hedge $h \in S$, and partial run v of $A_{e(S)}$ on h that ends in state sel , then $h \in \mathcal{L}(A)$.*

So, if the partial run reaches state sel then $\text{proj}_\Sigma(v)$ is certain for membership to $\mathcal{L}(A)$ with respect to S . Evaluating a $\text{SHA}^\downarrow_s A_{e(S)}$ yields an earliest in-memory membership tester for dSHA A . A single adaptation is as follows: whenever sel is reached, the evaluation can stop all over and accept the input hedge.

Theorem A1 (Completeness). *Let $A = (\Sigma, \mathcal{Q}, \Delta, I, F)$ be a dSHA that is schema-complete for schema $S = \mathcal{L}(A[F/F_S])$. For any hedge $h \in S$ and prefix v of $\text{nw}(h)$:*

- *If v is certain for membership in $\mathcal{L}(A)$ with S , then there exists a partial run w for $A_{e(S)}$ on h , such that $v = \text{proj}_\Sigma(w)$ and w ends with state sel .*
- *If v is certain for nonmembership in $\mathcal{L}(A)$ with S , then there exists blocking partial run w for $A_{e(S)}$ on h , such that $\text{proj}_\Sigma(w)$ is a prefix of v .*

This means that certainty for membership and nonmembership is detected at the earliest prefix when running the evaluator for $A_{e(S)}$.

Proof. We notice that this algorithm is basically the same as the earliest membership tester from Proposition 6 of [21], except that it also check for safe rejection and that it accounts for schemas. The fact that NWAs are used there instead of $\text{SHA}^\downarrow_s$ here is not essential. So, we can lift the soundness and completeness proof from there without problem. $\square$

Appendix B. Proofs for Section 6 (Safe-No-Change Projection)

Proof. We have to prove that no more changing states $q \in P \cup \text{no-change}^\Delta$ is sound.

If $q \in \text{no-change}^\Delta$, this follows from the schema-completeness of Δ so that one can neither block on any hedge from the schema nor change the state. In the case $q \in P$, the

intuition is that the state on level above, say r , can neither change, since $P = s\text{-no-change}^\Delta(r)$, nor can the automaton block on any hedge from the schema due to schema-completeness.

We first prove the inclusion $\mathcal{L}(A) \subseteq \mathcal{L}(A^{snc})$. Since $\mathcal{L}(A) \subseteq \mathbf{S}$ by definition of schemas, this implies $\mathcal{L}(A) \subseteq \mathcal{L}(A^{snc}) \cap \mathbf{S}$. The proof will be based on the following three Claims A1.1a, A1.2a, and A1.3a. Note that schema-completeness is not needed for this direction.

Claim A1.1a. $\Pi \xrightarrow{h} \Pi$ wrt Δ^{snc} for all hedges $h \in \mathcal{H}_\Sigma$.

The proof is straightforward by induction on the structure of h . It uses the last three transition rules of Δ^{snc} in Figure 10 permitting to always stay in Π for whatever hedge follows.

Claim A1.2a. For all $h \in \mathcal{H}_\Sigma$, $q \in \mathcal{Q}$, and $P \subseteq \mathcal{Q}$, such that $q \in P \cup \text{no-change}^\Delta$,

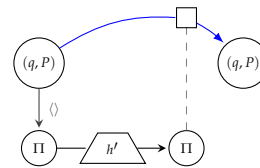
$$(q, P) \xrightarrow{h} (q, P) \text{ wrt } \Delta^{snc}.$$

We prove Claim A1.2a by induction on the structure of h .

Case $h = \langle h' \rangle$. In this case, we can use Claim A1.1a to show $\Pi \xrightarrow{h'} \Pi$ wrt Δ^{snc} and the inference rules

$$\frac{q \in P \cup \text{no-change}^\Delta}{(q, P) \xrightarrow{\langle \rangle} \Pi \text{ in } \Delta^{snc}} \quad \frac{q \in P \cup \text{no-change}^\Delta}{(q, P) @ \Pi \rightarrow (q, P) \text{ in } \Delta^{snc}}$$

in order to close the following diagram with respect to Δ^{snc} :



This proves $(q, P) \xrightarrow{h} (q, P)$ wrt Δ^{snc} , as required by the claim.

Case $h = a$. Since $q \in P \cup \text{no-change}^\Delta$ we can apply the inference rule:

$$\frac{a \in \Sigma \quad q \in P \cup \text{no-change}^\Delta}{(q, P) \xrightarrow{a} (q, P) \text{ in } \Delta^{snc}}$$

This proves this case of the claim.

Case $h = \varepsilon$. We trivially have $(q, P) \xrightarrow{\varepsilon} (q, P)$ wrt Δ^{snc} .

Case $h = h' \cdot h''$. By induction hypothesis applied to h' and h'' we have: $(q, P) \xrightarrow{h'} (q, P)$ and $(q, P) \xrightarrow{h''} (q, P)$ wrt Δ^{snc} . Hence, $(q, P) \xrightarrow{h' \cdot h''} (q, P)$ wrt Δ^{snc} .

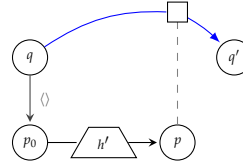
This ends the proof of Claim A1.2a. The next claim, in which the induction step is a little more tedious to prove, is the key of the soundness proof. We define the following predicate for all $q', q'' \in \mathcal{Q}$ and $P \subseteq \mathcal{Q}$:

$$q' \sim_P q'' \text{ iff } (q' = q'' \vee q', q'' \in P)$$

Claim A1.3a. Let $h \in \mathcal{H}_\Sigma$ be a hedge, $q, q' \in \mathcal{Q}$ states and $P \subseteq \mathcal{Q}$ a subset of states, such that $\text{acc}^\Delta(P) \subseteq P$ and $q \notin P \cup \text{no-change}^\Delta$. If $q \xrightarrow{h} q'$ wrt Δ , then there exists q'' , such that $(q, P) \xrightarrow{h} (q'', P)$ wrt Δ^{snc} and $q' \sim_P q''$.

Proof. Induction on the structure of h .

Case $h = \langle h' \rangle$. The assumption $q \xrightarrow{h} q'$ wrt Δ shows that there exists states $p_0 \in \langle \rangle^\Delta$ and $p \in Q$ closing the following diagram:

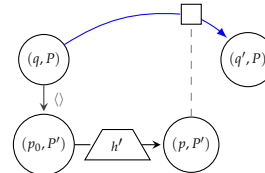


Let $P' = s\text{-no-change}^\Delta(q)$ and note that $\text{acc}^\Delta(P') \subseteq P'$. Since $q \notin P \cup \text{no-change}^\Delta$ we can infer the following:

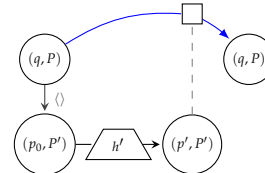
$$\frac{\langle \rangle \xrightarrow{\langle \rangle} p_0 \text{ in } \Delta \quad q \notin P \cup \text{no-change}^\Delta}{(q, P) \xrightarrow{\langle \rangle} (p_0, P') \text{ in } \Delta^{snc}} \quad \frac{q@p \rightarrow q' \text{ in } \Delta \quad q \notin P \cup \text{no-change}^\Delta}{(q, P)@(p, P') \rightarrow (q', P) \text{ in } \Delta^{snc}}$$

Subcase $p_0 \notin P' \cup \text{no-change}^\Delta$. The induction hypothesis applied to h' shows that there exists p' , such that $(p_0, P') \xrightarrow{h'} (p', P')$ wrt Δ^{snc} and $p \sim_P p'$. We distinguish the two cases justifying the latter predicate:

Subsubcase $p' = p$. Hence, $(p_0, P') \xrightarrow{h'} (p, P')$ wrt Δ^{snc} , so we can close the diagram as follows:



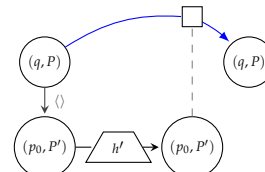
This shows that $(q, P) \xrightarrow{h} (q', P)$ wrt Δ^{snc} , and thus, the claim since $q' \sim_P q'$.
Subsubcase $p, p' \in P'$. Since $p' \in P'$ and $P' = s\text{-no-change}^\Delta(q)$, we have $q@p' \rightarrow q$ in Δ . Hence, we can close the diagram as follows:



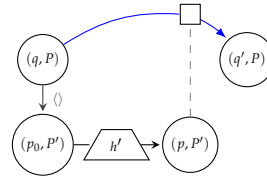
Since $p \in P'$ and $q@p \rightarrow q'$ in Δ , we have $q = q'$ by definition of $P' = s\text{-no-change}^\Delta(q)$. This shows that $(q, P) \xrightarrow{h} (q, P)$ wrt Δ^{snc} . Since $q \sim_P q$, the claim follows.

Subcase $p_0 \in P' \cup \text{no-change}^\Delta$. Claim A1.2a then shows that $(p_0, P') \xrightarrow{h'} (p_0, P')$ wrt Δ^{snc} .

Subsubcase $p_0 \in P'$. Since $p \in \text{acc}^\Delta(p_0)$ and $\text{acc}^\Delta(P') \subseteq P'$, it follows that $p \in P'$, too. By definition $P' = s\text{-no-change}^\Delta(q)$ and the completeness of Δ , the memberships $p_0 \in P'$ and $p \in P'$ imply that $q@^\Delta p_0 = \{q\} = q@^\Delta p$. We can now close the diagram below as follows:



Subsubcase $p_0 \in \text{no-change}^\Delta$. In this case, $p_0 = p$ so that $q' \in q@^\Delta p_0$. Hence,



Case $h = a$. Since $q \notin P \cup \text{no-change}^\Delta$, we can apply the inference rule:

$$\frac{q \xrightarrow{a} q' \text{ in } \Delta \quad q \notin P \cup \text{no-change}^\Delta}{(q, P) \xrightarrow{a} (q', P) \text{ in } \Delta^{snc}}$$

This shows that $(q, P) \xrightarrow{h} (q', P)$ validates the claim since $q' \sim_P q'$.

Case $h = \varepsilon$. In this case, we have $q = q'$ and $(q, P) \xrightarrow{\varepsilon} (q, P)$ so the claim holds.

Case $h = h_1 \cdot h_2$. Since $q \xrightarrow{h} q'$ wrt Δ , there exists $q_1 \in \mathcal{Q}$, such that $q \xrightarrow{h_1} q_1$ wrt Δ and $q_1 \xrightarrow{h_2} q'$ wrt Δ . Since $q \notin P \cup \text{no-change}^\Delta$, we apply the induction hypothesis on h_1 . This implies that there exists q'_1 such that,

$$(q, P) \xrightarrow{h_1} (q'_1, P) \text{ wrt } \Delta^{snc} \text{ and } q_1 \sim_P q'_1.$$

We distinguish the two cases of $q_1 \sim_P q'_1$:

Subcase $q_1 = q'_1$. We also distinguish two subcases here:

Subsubcase $q_1 \notin P$. The induction hypothesis applied to h_2 yields the following:

$$\exists q'' . (q_1, P) \xrightarrow{h_2} (q'', P) \text{ wrt } \Delta^{snc} \wedge q' \sim_P q'',$$

Hence

$$\exists q'' . (q, P) \xrightarrow{h} (q'', P) \text{ wrt } \Delta^{snc} \wedge q' \sim_P q''$$

Subsubcase $q_1 \in P$. By Claim A1.2a, we have $(q_1, P) \xrightarrow{h_2} (q_1, P)$. We also have $q' \in \text{acc}(\{q_1\})$, and since we assume $\text{acc}(P) \subseteq P$, this implies $q' \in P$. Hence, $(q, P) \xrightarrow{h} (q_1, P)$ and $q', q_1 \in P$, implying the claim is $q' \sim_P q_1$.

Subcase $q_1, q'_1 \in P$. Since $q'_1 \in P$, Claim A1.2a implies $(q'_1, P) \xrightarrow{h_2} (q'_1, P)$ wrt Δ^{snc} . Thus, $(q, P) \xrightarrow{h} (q'_1, P)$ wrt Δ^{snc} . Since $q' \in \text{acc}^\Delta(\{q_1\})$ and $q_1 \in P$, it follows that $q' \in \text{acc}^\Delta(P) \subseteq P$. Here, we used as in the previous subsubcase that $\text{acc}(P) \subseteq P$ is assumed by the claim. Let $q'' = q'_1$. Then, we have $(q, P) \xrightarrow{h} (q'', P)$ wrt Δ^{snc} and $q', q'' \in P$ showing the claim.

This ends the proof of Claim A1.3a. $\square$

Proof of Inclusion $\mathcal{L}(A) \subseteq \mathcal{L}(A^{snc})$. Let $h \in \mathcal{L}(A)$. Then, there exists $q_0 \in I$ and $q \in F$, such that $q_0 \xrightarrow{h} q$. We distinguish two cases:

Case $q_0 \in \text{no-change}^\Delta$. By definition of no-change^Δ and since $q \in \text{acc}^\Delta(q_0)$ we have $q_0 = q$.

Claim A1.2a shows that $(q_0, \emptyset) \xrightarrow{h} (q_0, \emptyset)$ wrt Δ^{snc} and thus $(q_0, \emptyset) \xrightarrow{h} (q, \emptyset)$ so that $h \in \mathcal{L}(A^{snc})$.

Case $q_0 \notin \text{no-change}^\Delta$. Claim A1.3a with $P = \emptyset$ shows that $(q_0, \emptyset) \xrightarrow{h} (q, \emptyset)$ wrt Δ^{snc} and hence $h \in \mathcal{L}(A^{snc})$.

This ends the proof of the first inclusion. We next want to show the inverse inclusion $\mathcal{L}(A^{snc}) \cap \mathbf{S} \subseteq \mathcal{L}(A)$. It will eventually follow from the following three Claims A1.1b, A1.2b, and A1.3b.

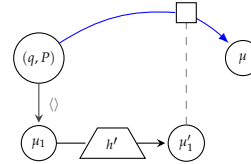
Claim A1.1b. For any hedge h and state $\mu \in \mathcal{Q}^{snc}$, if $\Pi \xrightarrow{h} \mu$ wrt Δ^{snc} then $\mu = \Pi$.

The proof is straightforward by induction on the structure of h : the only transitions rules of Δ^{snc} with Π on the left hand side are inferred by the last three rules in Figure 10. These require to stay in Π whatever hedge follows.

Claim A1.2b. For any hedge h , set $P \subseteq \mathcal{Q}$, state $q \in P \cup \text{no-change}^\Delta$, and state $\mu \in \mathcal{Q}^{snc}$: if $(q, P) \xrightarrow{h} \mu$ wrt Δ^{snc} , then $\mu = (q, P)$.

Proof. By induction on the structure of h . Suppose that $(q, P) \xrightarrow{h} \mu$ wrt Δ^{snc} .

Case $h = \langle h' \rangle$. There must exist states $\mu_1, \mu'_1 \in \mathcal{Q}^{snc}$ closing the following diagram:



Since $q \in P \cup \text{no-change}^\Delta$, the following rule must have been applied to infer $(q, P) \xrightarrow{\langle \rangle} \mu_1$ wrt Δ^{snc} :

$$\frac{q \in P \cup \text{no-change}^\Delta}{(q, P) \xrightarrow{\langle \rangle} \Pi \text{ in } \Delta^{snc}}$$

Therefore, $\mu_1 = \Pi$. Claim A1.1b shows that $\mu'_1 = \Pi$, too. So, μ must have been inferred by applying the rule:

$$\frac{q \in P \cup \text{no-change}^\Delta}{(q, P) @ \Pi \rightarrow (q, P) \text{ in } \Delta^{snc}}.$$

So, $\mu = (q, P)$, as required.

Case $h = a$. The following rule must have been applied:

$$\frac{q \in P \cup \text{no-change}^\Delta}{(q, P) \xrightarrow{a} (q, P) \text{ in } \Delta^{snc}}$$

Hence, $\mu = (q, P)$.

Case $h = \varepsilon$. Obvious.

Case $h = h_1 \cdot h_2$. There must exist μ_1 , such that $(q, P) \xrightarrow{h_1} \mu_1 \xrightarrow{h_2} \mu$ wrt Δ^{snc} . By induction hypothesis applied to h_1 , we have $\mu_1 = (q, P)$. We can thus apply the induction hypothesis to h_2 to obtain $\mu_2 = (q, P)$.

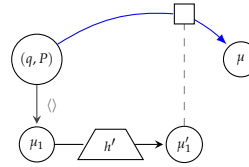
This ends the proof of Claim A1.2b. We next need an inverse of Claim A1.3a. $\square$

Claim A1.3b. Let $q \in \mathcal{Q}$, $P \subseteq \mathcal{Q}$, such that $q \notin P \cup \text{no-change}^\Delta$ and $\text{acc}^\Delta(P) \subseteq P$. For any $h \in \mathcal{H}_{\Sigma}$, such that $(q, P) \xrightarrow{h} \mu$ wrt Δ^{snc} for some $\mu \in \mathcal{Q}^{snc}$ and such that Δ does not have any blocking partial run on h starting from q , there exists q', q'' , such that

$$\mu = (q', P), q \xrightarrow{h} q'' \text{ wrt } \Delta, \text{ and } q' \sim_P q''.$$

Proof. By induction on the structure of $h \in \mathcal{H}_\Sigma$, we distinguish cases for all possible forms h .

Case $h = \langle h' \rangle$. By definition of $(q, P) \xrightarrow{h} \mu$ wrt Δ^{snc} , there must exist $\mu_1, \mu'_1 \in \mathcal{Q}^{snc}$, such that the following diagram can be closed:



Since $q \notin P \cup no-change^\Delta$, the following rule got applied to infer $(q, P) \xrightarrow{\langle \rangle} \mu_1$ where $P' = s-no-change^\Delta(q)$:

$$\frac{\xrightarrow{\langle \rangle} p \text{ in } \Delta \quad q \notin P \cup no-change^\Delta}{(q, P) \xrightarrow{\langle \rangle} (p, P') \text{ in } \Delta^{snc}}$$

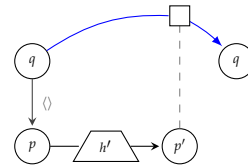
Hence, there exists $p \in \langle \rangle^\Delta$, such that $\mu_1 = (p, P')$. We fix such a state p arbitrarily.

Since Δ does not have any blocking runs on h from q there exists $p', q'' \in \mathcal{Q}$, such that $p \xrightarrow{h'} p'$ and $q@p' \rightarrow q''$ wrt Δ . Furthermore, Δ does not have any blocking partial run on h' starting from p .

Subcase $p \in P'$. In this case, we can apply Claim A1.2b to $(p, P') \xrightarrow{h'} \mu'_1$ wrt Δ in order to show that $\mu'_1 = (p, P')$. Since $p \in acc^\Delta(\{p\})$ and $p \in P' = s-no-change^\Delta(q)$ it follows that $q@^\Delta p = \{q\}$. Hence, the following rule got applied to infer $(q, P) \xrightarrow{h} \mu$:

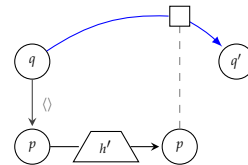
$$\frac{q@p \rightarrow q \text{ in } \Delta \quad q \notin P \cup no-change^\Delta}{(q, P)@(p, P') \rightarrow (q, P) \text{ in } \Delta^{snc}},$$

This shows that $\mu = (q, P)$. Let $q' = q$ so that $\mu = (q', P)$. Since $p \xrightarrow{h'} p'$ wrt Δ , we have $p' \in acc^\Delta(\{p\})$ so that $q@p' \rightarrow q$ wrt Δ by definition of $s-no-change^\Delta(\{q\})$. Hence, we can close the following diagram:



Let $q'' = q$, so that $q' = q''$. It then holds that $q' \sim_P q''$ and $q \xrightarrow{\langle h' \rangle} q'$ wrt Δ , as required by the claim.

Subcase $p \in no-change^\Delta$. In this case $p \xrightarrow{h'} p'$ wrt Δ implies that $p' = p$. Hence, we can close the following diagram:



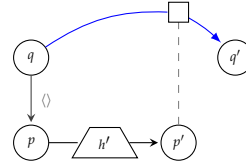
This shows that $q \xrightarrow{\langle h' \rangle} q'$ wrt Δ .

Subcase $p \notin P' \cup \text{no-change}^\Delta$. By induction hypothesis applied to $(p, P') \xrightarrow{h'} \mu'_1$ wrt Δ , there exists p'' , such that

$$\mu'_1 = (p', P'), p \xrightarrow{h'} p'' \text{ wrt } \Delta, \text{ and } p' \sim_{P'} p''.$$

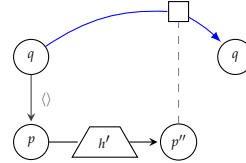
Since Δ does not have blocking runs on h starting with q , there exist q'' such that $q@p'' \rightarrow q''$ wrt Δ . There are two ways to satisfy $p' \sim_{P'} p''$:

Subsubcase $p' = p''$. We then have the following:



This shows $q \xrightarrow{\langle h' \rangle} q'$ wrt Δ .

Subsubcase $p', p'' \in P$. By definition of P' , it follows that $q' = q = q''$. Hence,



This shows $q \xrightarrow{\langle h' \rangle} q$ wrt Δ .

Case $h = a$. Since $q \notin P \cup \text{no-change}^\Delta$, the following inference rule must be used:

$$\frac{q \xrightarrow{a} q' \text{ in } \Delta \quad q \notin P \cup \text{no-change}^\Delta}{(q, P) \xrightarrow{a} (q', P) \text{ in } \Delta^{snc}}$$

So $\mu = (q', P)$ and $q \xrightarrow{a} q'$ wrt Δ .

Case $h = \varepsilon$. Obvious.

Case $h = h_1 \cdot h_2$. The judgement $(q, P) \xrightarrow{h} \mu$ wrt Δ^{snc} shows that there exists μ_1 , such that $(q, P) \xrightarrow{h_1} \mu_1 \xrightarrow{h_2} \mu$ wrt Δ^{snc} . Since $q \notin P \cup \text{no-change}^\Delta$, we can apply the induction hypothesis to h_1 . It shows that there exists q'_1 and q''_1 , such that $\mu_1 = (q'_1, P)$, $q \xrightarrow{h_1} q'_1$ wrt Δ , and $q'_1 \sim_P q''_1$.

Subcase $q'_1 = q''_1$. Hence, $(q_1, P) \xrightarrow{h_2} \mu$ wrt Δ^{snc} .

Subsubcase $q'_1 \notin P \cup \text{no-change}^\Delta$. In this case, we can apply the induction hypothesis to $(q'_1, P) \xrightarrow{h_2} \mu$ wrt Δ^{snc} , showing the existence of q' , such that $\mu = (q', P)$ and state q'' , such that $q'_1 \xrightarrow{h_2} q''$ and $q' \sim_P q''$. Hence, $\exists q'' . q \xrightarrow{h} q''$ wrt Δ and $q' \sim_P q''$ so the claim holds.

Subsubcase $q'_1 \in P \cup \text{no-change}^\Delta$. Claim A1.2b applied to $(q'_1, P) \xrightarrow{h_2} \mu$ wrt Δ^{snc} shows that $\mu = (q'_1, P)$.

Subcase $q'_1, q''_1 \in P$. Recall that $(q''_1, P) \xrightarrow{h_2} \mu$ wrt Δ^{snc} and $q''_1 \in P$. Claim A1.2b shows that $\mu = (q''_1, P)$ wrt Δ^{snc} . We also have $q \xrightarrow{h_1} q'_1$ wrt Δ . Since there are no blocking partial runs on h starting from q there exist a state q'' such that $q'_1 \xrightarrow{h_2} q''$ wrt Δ . Since $q'_1 \in P$ and P is closed by accessibility, we have $q'' \in \text{acc}(\{q''_1\}) \subseteq \text{acc}(P) \subseteq P$. From $q \xrightarrow{h_1} q'_1$ wrt Δ , we get $q \xrightarrow{h} q''$ wrt Δ . Since $q''_1, q'' \in P$, it follows that $q''_1 \sim_P q''$, and thus, the claim holds.

This ends the proof of Claim A1.3b.

□

Proof of Inclusion $\mathcal{L}(A^{snc}) \cap \mathbf{S} \subseteq \mathcal{L}(A)$. Let $h \in \mathcal{L}(A^{snc}) \cap \mathbf{S}$. Since $h \in \mathcal{L}(A^{snc})$, then there exists $q_0 \in I$ and $q \in F$, such that $(q_0, \emptyset) \xrightarrow{h} (q, \emptyset)$ wrt Δ^{snc} . Using Lemma 7, we have $q_0 \xrightarrow{h} q$ wrt Δ .

We distinguish two cases:

Case $q_0 \in \text{no-change}^\Delta$. Claim A1.2b shows that $q = q_0$. Since A is schema-complete for $\mathbf{S}$, $h \in \mathbf{S}$, and $q_0 \in I \cap \text{no-change}^\Delta$, Lemma 7 shows that $q_0 \xrightarrow{h} q_0$ wrt Δ . Since $q = q_0$, this yields $h \in \mathcal{L}(A)$.

Case $q_0 \notin \text{no-change}^\Delta$. Since A is schema-complete for $\mathbf{S}$ and $h \in \mathbf{S}$ there exist no blocking runs on h that start in q_0 . Therefore, we can apply Claim A1.3b with $P = \emptyset$ to $(q_0, \emptyset) \xrightarrow{h} (q, \emptyset)$ wrt Δ^{snc} . This shows that $q_0 \xrightarrow{h} q$ wrt Δ , and hence, $h \in \mathcal{L}(A)$.

This ends the proof of the inverse inclusion, and thus, of $\mathcal{L}(A) = \mathcal{L}(A^{snc})$. □

References

1. Marian, A.; Siméon, J. Projecting XML Documents. In *Proceedings of the VLDB*; Morgan Kaufmann: San Francisco, CA, USA, 2003; pp. 213–224.
2. Frisch, A. Regular Tree Language Recognition with Static Information. In *Proceedings of the Exploring New Frontiers of Theoretical Informatics, IFIP 18th World Computer Congress, TCS 3rd International Conference on Theoretical Computer Science*, Toulouse, France, 22–27 August 2004; pp. 661–674.
3. Maneth, S.; Nguyen, K. XPath Whole Query Optimization. *VLPB J.* **2010**, *3*, 882–893.
4. Sebastian, T.; Niehren, J. Projection for Nested Word Automata Speeds up XPath Evaluation on XML Streams. In *Proceedings of the International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM)*, Harrachov, Czech Republic, 23–28 January 2016.
5. Kay, M. The Saxon XSLT and XQuery Processor. 2004. Available online: <https://www.saxonica.com> (accessed on 29 July 2024).
6. Gienieccko, M.; Murlak, F.; Paperman, C. Supporting Descendants in SIMD-Accelerated JSONPath. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, La Jolla, CA, USA, 27 April–1 May 2024; Volume 1, pp. 15–29.
7. Al Serhali, A.; Niehren, J. A Benchmark Collection of Deterministic Automata for XPath Queries. In *Proceedings of the XML Prague 2022*, Prague, Czech Republic, 9–11 June 2022.
8. Niehren, J.; Sakho, M. Determinization and Minimization of Automata for Nested Words Revisited. *Algorithms* **2021**, *14*, 68. <https://doi.org/10.3390/a14030068>.
9. Comon, H.; Dauchet, M.; Gilleron, R.; Löding, C.; Jacquemard, F.; Lugiez, D.; Tison, S.; Tommasi, M. Tree Automata Techniques and Applications. Available online: <http://tata.gforge.inria.fr> (accessed on 29 July 2024).
10. Thatcher, J.W. Characterizing derivation trees of context-free grammars through a generalization of automata theory. *J. Comput. Syst. Sci.* **1967**, *1*, 317–322. [https://doi.org/10.1016/S0022-0000\(67\)80022-9](https://doi.org/10.1016/S0022-0000(67)80022-9).
11. Alur, R. Marrying Words and Trees. In *Proceedings of the 26th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, Beijing, China, 11–14 June 2007; ACM-Press: New York, NY, USA, 2007; pp. 233–242.
12. Alur, R.; Madhusudan, P. Visibly pushdown languages. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, Chicago, IL, USA, 13–16 June 2004; Babai, L., Ed.; ACM: New York, NY, USA, 2004; pp. 202–211. <https://doi.org/10.1145/1007352.1007390>.
13. Mehlhorn, K. Pebbling Mountain Ranges and its Application of DCFL-Recognition. In *Proceedings of the Automata, Languages and Programming, 7th Colloquium*, Noordwijkerhout, The Netherlands, 14–18 July 1980; *Proceedings*; de Bakker, J.W., van Leeuwen, J., Eds.; Springer: Berlin/Heidelberg, Germany, 1980; Volume 85, pp. 422–435. https://doi.org/10.1007/3-540-10003-2_89.
14. von Braunmühl, B.; Verbeek, R. Input Driven Languages are Recognized in log n Space. In *Topics in the Theory of Computation; North-Holland Mathematics Studies*; Karpinski, M., van Leeuwen, J., Eds.; North-Holland: 1985; Volume 102, pp. 1–19. [https://doi.org/10.1016/S0304-0208\(08\)73072-X](https://doi.org/10.1016/S0304-0208(08)73072-X).
15. Okhotin, A.; Salomaa, K. Complexity of input-driven pushdown automata. *SIGACT News* **2014**, *45*, 47–67. <https://doi.org/10.1145/2636805.2636821>.
16. Niehren, J.; Sakho, M.; Al Serhali, A. Schema-Based Automata Determinization. In *Proceedings of the 13th International Symposium on Games, Automata, Logics and Formal Verification, GandALF 2022*, Madrid, Spain, 21–23 September 2022; Ganty, P., Monica, D.D., Eds.; EPTCS: 2022; Volume 370, pp. 49–65. <https://doi.org/10.4204/EPTCS.370.4>.
17. Lick, A.; Schmitz, S. XPath Benchmark. Available online: <https://archive.softwareheritage.org/browse/directory/1ea68cf5bb3f9f3f2fe8c7995f1802ebadf17fb5> (accessed on 29 July 2024).

18. Mozafari, B.; Zeng, K.; Zaniolo, C. High-performance complex event processing over XML streams. In *Proceedings of the SIGMOD Conference*; Candan, K.S., Chen, Y., Snodgrass, R.T., Gravano, L., Fuxman, A., Candan, K.S., Chen, Y., Snodgrass, R.T., Gravano, L., Fuxman, A., Eds.; ACM: New York, NY, USA, 2012; pp. 253–264. <https://doi.org/10.1145/2213836.2213866>.
19. Grez, A.; Riveros, C.; Ugarte, M. A Formal Framework for Complex Event Processing. In *Proceedings of the 22nd International Conference on Database Theory (ICDT 2019)*, Lisbon, Portugal, 26–28 March 2019; pp. 5:1–5:18. <https://doi.org/10.4230/LIPIcs.ICDT.2019.5>.
20. Muñoz, M.; Riveros, C. Streaming Enumeration on Nested Documents. In *Proceedings of the 25th International Conference on Database Theory, ICDT 2022*, Edinburgh, UK, 29 March–1 April 2022; Olteanu, D., Vortmeier, N., Eds.; Schloss Dagstuhl - Leibniz-Zentrum für Informatik: 2022; Volume 220, pp. 19:1–19:18. <https://doi.org/10.4230/LIPIcs.ICDT.2022.19>.
21. Al Serhali, A.; Niehren, J. Earliest Query Answering for Deterministic Stepwise Hedge Automata. In *Proceedings of the Implementation and Application of Automata—27th International Conference, CIAA 2023*, Famagusta, North Cyprus, 19–22 September 2023; Proceedings; Nagy, B., Ed.; Springer: Berlin/Heidelberg, Germany, 2023; Volume 14151, pp. 53–65. https://doi.org/10.1007/978-3-031-40247-0_3.
22. Gauwin, O.; Niehren, J.; Tison, S. Earliest Query Answering for Deterministic Nested Word Automata. In *Proceedings of the 17th International Symposium on Fundamentals of Computer Theory*, Wroclaw, Poland, 2–4 September 2009; Springer: Berlin/Heidelberg, Germany, 2009; Volume 5699, pp. 121–132.
23. Debarbieux, D.; Gauwin, O.; Niehren, J.; Sebastian, T.; Zergaoui, M. Early nested word automata for XPath query answering on XML streams. *Theor. Comput. Sci.* **2015**, *578*, 100–125. <https://doi.org/10.1016/j.tcs.2015.01.017>.
24. Al Serhali, A.; Niehren, J. Subhedge Projection for Stepwise Hedge Automata. In *Proceedings of the Fundamentals of Computation Theory—24th International Symposium, FCT 2023*, Trier, Germany, 18–21 September 2023; Proceedings; Fernau, H., Jansen, K., Eds.; Springer: Berlin/Heidelberg, Germany, 2023; Volume 14292, pp. 16–31. https://doi.org/10.1007/978-3-031-43587-4_2.
25. Neumann, A.; Seidl, H. Locating Matches of Tree Patterns in Forests. In *Proceedings of the Foundations of Software Technology and Theoretical Computer Science*, Chennai, India, 17–19 December 1998; Springer: Berlin/Heidelberg, Germany, 1998; Volume 1530, pp. 134–145.
26. Gauwin, O.; Niehren, J.; Roos, Y. Streaming Tree Automata. *Inf. Process. Lett.* **2008**, *109*, 13–17. <https://doi.org/10.1016/j.ipl.2008.08.002>.
27. Franceschet, M. XPathMark Performance Test. Available online: <https://users.dimi.uniud.it/~massimo.franceschet/xpathmark/PTbench.html> (accessed on 25 October 2020).
28. Hosoya, H.; Pierce, B.C. XDuce: A Statically Typed XML Processing Language. *ACM Trans. Internet Technol.* **2003**, *3*, 117–148. <https://doi.org/10.1145/767193.767195>.
29. Gécseg, F.; Steinby, M. *Tree Automata*; Akadémiai Kiadó: Budapest, Hungary, 1984.
30. Gold, E.M. Language Identification in the Limit. *Inf. Control.* **1967**, *10*, 447–474. [https://doi.org/10.1016/S0019-9958\(67\)91165-5](https://doi.org/10.1016/S0019-9958(67)91165-5).
31. Carme, J.; Niehren, J.; Tommasi, M. Querying Unranked Trees with Stepwise Tree Automata. In *Proceedings of the Rewriting Techniques and Applications, 15th International Conference, RTA 2004*, Aachen, Germany, 3–5 June 2004; Proceedings; van Oostrom, V., Ed.; Springer: Berlin/Heidelberg, Germany, 2004; Volume 3091, pp. 105–118. https://doi.org/10.1007/978-3-540-25979-4_8.
32. Pair, C.; Quéré, A. Définition et étude des langages réguliers. *Inf. Control* **1968**, *13*, 565–593.
33. Murata, M. Hedge Automata: A Formal Model for XML Schemata. 2000. Available online: http://www.xml.gr.jp/relax/hedge_nice.html (accessed on 29 July 2024).
34. Martens, W.; Niehren, J. On the Minimization of XML Schemas and Tree Automata for Unranked Trees. *J. Comput. Syst. Sci.* **2007**, *73*, 550–583.
35. Niehren, J. Stepwise Hedge Automata are exponentially more succinct than Forest Automata. 2024, *in preparation*.
36. Bojanczyk, M.; Walukiewicz, I. Forest algebras. In *Proceedings of the Logic and Automata: History and Perspectives [in Honor of Wolfgang Thomas]*; Flum, J., Grädel, E., Wilke, T., Eds.; Amsterdam University Press: Amsterdam, The Netherlands, 2008; Volume 2, pp. 107–132.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.

Article

Hierarchical Optimization Framework for Layout Design of Star-Tree Gas-Gathering Pipeline Network in Discrete Spaces

Yu Lin ¹, Yanhua Qiu ¹, Hao Chen ², Jun Zhou ^{3,*}, Jiayi He ³, Penghua Du ³ and Dafan Liu ⁴

¹ Research Institute of Gathering and Transportation Engineering Technology, PetroChina Southwest Oil & Gasfield Company, Chengdu 610041, China; linyu223@petrochina.com.cn (Y.L.); qiuyan0320@petrochina.com.cn (Y.Q.)

² Sichuan Shale Gas Exploration and Development Co., Ltd., Chengdu 610051, China; chen hao05@petrochina.com.cn

³ Petroleum Engineering School, Southwest Petroleum University, Chengdu 610500, China; hjywyx@163.com (J.H.); scdupenghua@163.com (P.D.)

⁴ Shale Gas Exploration and Development Department, CNPC Chuanqing Drilling Engineering Co., Ltd., Chengdu 610066, China; liudf_cqyyq@cnpc.com.cn

* Correspondence: zhoujunsupu@163.com

Abstract: The gas-gathering pipeline network is a critical infrastructure for collecting and conveying natural gas from the extraction site to the processing facility. This paper introduces a design optimization model for a star-tree gas-gathering pipeline network within a discrete space, aimed at determining the optimal configuration of this infrastructure. The objective is to reduce the investment required to build the network. Key decision variables include the locations of stations, the plant location, the connections between wells and stations, and the interconnections between stations. Several equality and inequality constraints are formulated, primarily addressing the affiliation between wells and stations, the transmission radius, and the capacity of the stations. The design of a star-tree pipeline network represents a complex, non-deterministic polynomial (NP) hard combinatorial optimization problem. To tackle this challenge, a hierarchical optimization framework coupled with an improved genetic algorithm (IGA) is proposed. The efficacy of the genetic algorithm is validated through testing and comparison with other traditional algorithms. Subsequently, the optimization model and solution methodology are applied to the layout design of a pipeline network. The findings reveal that the optimized network configuration reduces investment costs by 16% compared to the original design. Furthermore, when comparing the optimal layout under a star-star topology, it is observed that the investment needed for the star-star topology is 4% higher than that needed for the star-tree topology.

Keywords: topological structure; layout design; hierarchical optimization; genetic algorithm



Citation: Lin, Y.; Qiu, Y.; Chen, H.; Zhou, J.; He, J.; Du, P.; Liu, D. Hierarchical Optimization Framework for Layout Design of Star-Tree Gas-Gathering Pipeline Network in Discrete Spaces. *Algorithms* **2024**, *17*, 340. <https://doi.org/10.3390/a17080340>

Academic Editors: Frank Werner, Ming-Feng Ge and Chen Liu

Received: 3 July 2024

Revised: 27 July 2024

Accepted: 3 August 2024

Published: 5 August 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

1.1. Motivation

Currently, the global community is transitioning into an era characterized by clean and efficient energy, with natural gas emerging as the predominant energy source. According to the Global Energy Outlook, worldwide consumption of natural gas is projected to increase continuously until 2050, with an average annual growth rate exceeding 1.4% (Richard, 2020). The significant rise in natural gas demand necessitates enhanced efforts toward the construction of surface gas-gathering pipeline networks. The investment required for gas-gathering pipeline networks is substantial, often comprising over 30% of the total development costs of a gas field [1]. Consequently, optimizing the design of these pipeline networks in a rational manner can yield considerable economic benefits.

The system typically includes wells, gas-gathering stations, central treatment facilities, and pipelines (Figure 1). Initially, the natural gas extracted from the wells is conveyed to

the gas-gathering station via production pipelines. Once collected at the gas-gathering station, the gas is then transported through gathering pipelines to the central treatment plant for purification, ensuring it meets the standards required for external distribution. Based on network topology, gas-gathering pipeline networks can be categorized into three fundamental types: star, tree, and cyclic topologies [2]. In a star topology, multiple wells are directly connected to a single station, as illustrated in Figure 2a. In a tree topology, wells are linked to the station through adjacent wells, as depicted in Figure 2b. A cyclic topology involves interconnected wells forming a loop, as shown in Figure 2c.

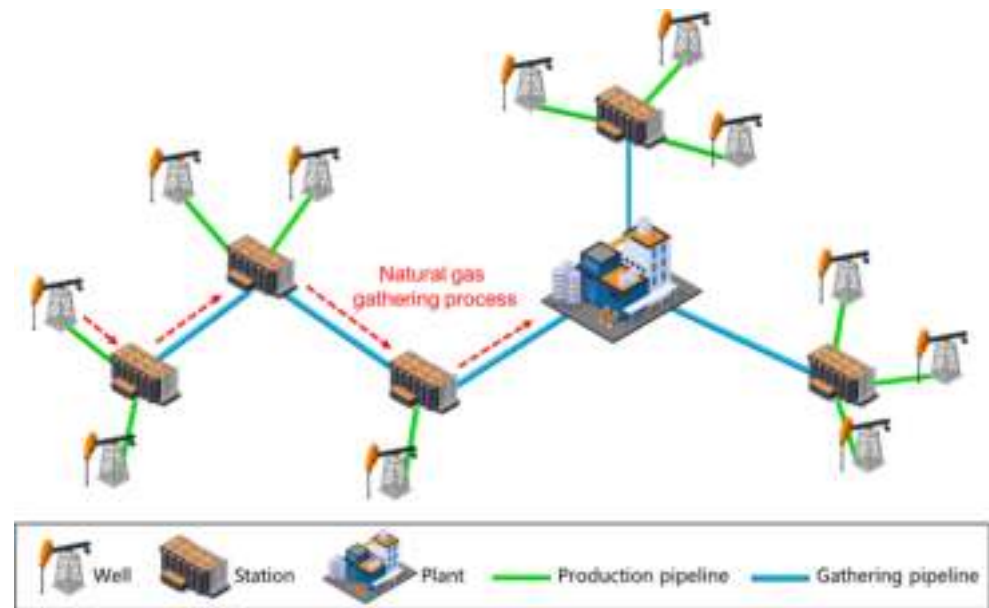


Figure 1. Typical gas-gathering pipeline network.

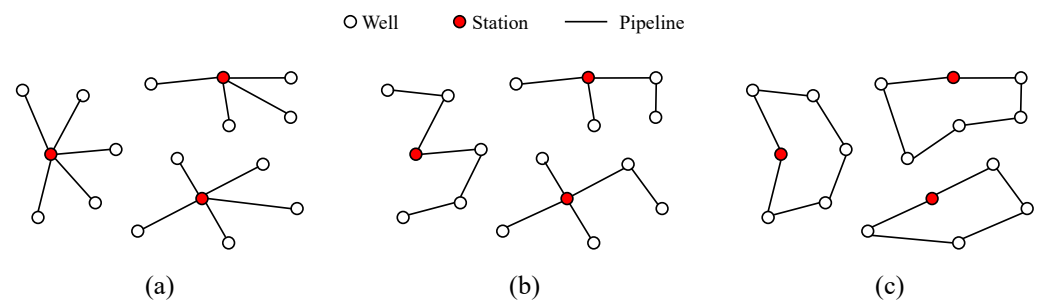


Figure 2. Pipeline network structure: (a) star topology, (b) tree topology, (c) cyclic topology.

In practical large-scale gas-gathering pipeline networks, various topological combinations are often observed. At present, in the actual construction of gas fields, based on a variety of factors, including gas well distribution, construction cost, management convenience, operation efficiency, etc., star–star topology network structures and star–tree topology network structures have been widely used in gas field development. The star–star topology network structure of multiple gas wells directly connected to a central gas-gathering site makes the layout simple and easy to manage and control. Its advantages are that the total length of the pipeline can be reduced, the initial construction cost can be reduced, and it is convenient to centrally process and transport the natural gas produced by the gas field, which is suitable for a gas field with a small gas-bearing area, a large number of wellheads, and a low output of a single well. The star–tree topology network structure combines the characteristics of the star and dendritic structures. In this structure, each gas well and its gas-collecting stations are connected by a star shape, and the gas-collecting stations are connected by gas-collecting trunk lines to form a dendritic structure. This lay-

out balances the cost and efficiency of the long-haul collection line, while providing some flexibility to adjust and expand according to the stage of development of the field. Figure 3 illustrates a typical star-tree gas-gathering pipeline network. This network structure is particularly suitable for scenarios where wells within local blocks are concentrated, but these blocks are widely dispersed with significant distances between them.

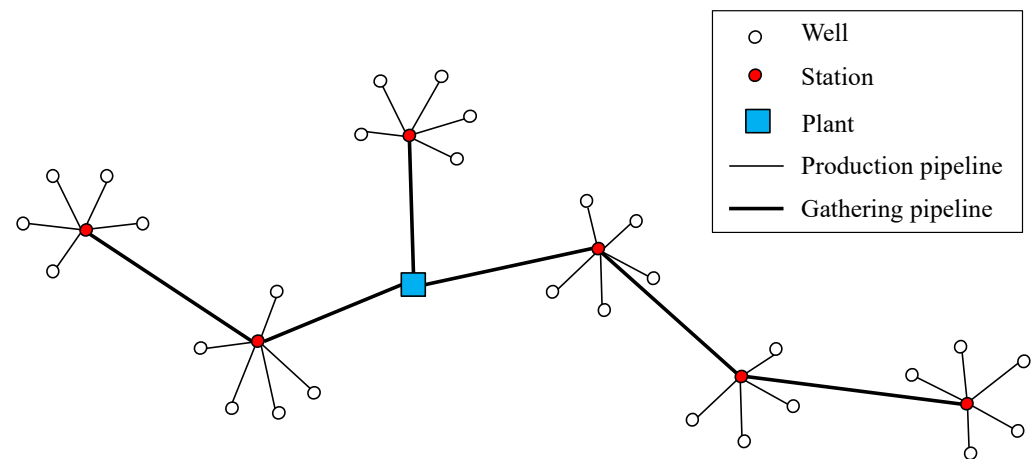


Figure 3. Star-tree pipeline network.

In general, the optimization problem can be categorized into design optimization and operational optimization [3]. Design optimization involves selecting the most suitable layout configuration, station placements, pipe sizes, and equipment scales for a new pipeline network to achieve optimal construction investment costs. Operational optimization, on the other hand, focuses on refining the operational strategies of the established pipeline network, which includes optimizing pipeline flows and node pressure parameters to meet technical or economic objectives.

The layout optimization challenge of gas-gathering pipeline networks encompasses several subprocesses, such as well grouping, station and plant location determination, and establishing connectivity among facilities [4]. Figure 4 depicts the layout optimization process. In addition, the determination of station (or plant) location is divided into two scenarios. The first scenario is that the station can be built at any location within the gas field. The second scenario is that the station must be jointly built with the well to save on construction investment costs. In other words, the station location can only be selected from the existing well location, which is defined as the station location selection in a discrete space. Since this scenario is more widely used in engineering practice, this paper will focus on this problem.

1.2. Literature Review

Based on literature research, it is evident that the academic community has conducted extensive studies on optimizing the design of gas pipeline networks. These studies primarily focus on modeling and solving optimization challenges related to network layout. Depending on the topology of the pipeline network—such as star, tree, and cyclic topologies—different optimization problems arise. Sanaye and Mahmoudimehr [5] developed an optimization model for tree topology gas pipeline networks, which considers parameters like network layout, pipe diameter, and node pressures. Rodriguez et al. [6] investigated the design of multi-level star networks, addressing optimal configurations and facility placements. Fu [7] explored the multi-objective optimization design of ring-type oil-gathering pipeline networks. Wang et al. [8] formulated a multi-period MILP (Mixed Integer Linear Programming) model for optimizing tree topology networks, accounting for terrain and obstacles. Zhou et al. [9] established an MINLP (Mixed Integer Nonlin-

ear Programming) model for optimizing two-level star topology pipeline networks for underground natural gas storage.

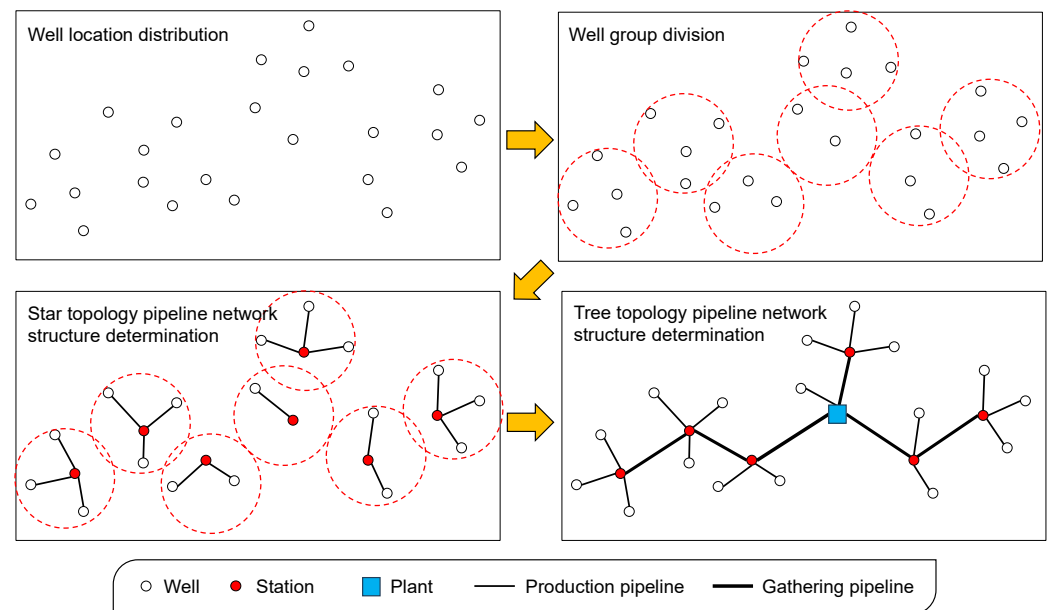


Figure 4. Hierarchical optimization diagram.

Regarding the optimization solving methods, early research predominantly utilized operation research techniques, particularly heuristic algorithms, to address the gas-gathering pipeline network optimization models. From a mathematical optimization and operation research perspective, designing optimal gas-gathering pipeline networks is classified as an NP-hard combinatorial optimization problem, which poses significant challenges and finds application across various industrial domains [10,11]. Balinski [12] formulated an optimization model for site placement in discrete spaces with unlimited site capacities, which was primarily solved using heuristic algorithms. Common exact algorithms include the branch and bound method, cutting plane algorithm, and column generation algorithm [13,14]. However, traditional optimization theories often struggle to efficiently solve problems with natural gas pipeline network models due to lengthy computational times and occasional mismatches between theoretical outcomes and practical applications.

In recent years, commercial solvers and heuristic algorithms have gradually become the most common way to deal with pipeline network layout optimization. A commercial solver can eliminate the tedious algorithm programming work, but the solution method is limited, and it is difficult to modify the built-in parameters of the algorithm. The heuristic algorithm has a large programming workload, but the algorithm has good scalability and can even combine the advantages of several algorithms to obtain the optimization results. Uester and Dilaveroglu [15] solved a gas pipeline network design and expansion optimization problem using the online Bonmin solver. Zhang et al. [16] developed a comprehensive MILP layout optimization model that incorporates terrain and obstacle considerations. There are also scholars who use graph theory methods and machine learning methods to solve optimization problems [17,18]. They utilized the Gurobi solver to globally solve the model. Rosa et al. [19] introduced a practical and efficient optimization approach for subsea production networks using the GAMS solver. Liu et al. [20] put forward an enhanced particle swarm optimization algorithm with enhanced global search capabilities to tackle the MINLP model for optimizing pipeline network layouts. Hong et al. [21] proposed a pipeline network layout optimization model that accounts for hydraulic characteristics and employed an ant colony algorithm for its solution.

1.3. Contributions

Gas field surface investment accounts for about 30% to 40% of the total development investment cost, and the proportion of investment in the construction of surface-level gathering and transportation pipeline networks often accounts for 60% to 70% of the surface investment cost. It can be seen that a natural gas-gathering and transmission pipeline network occupies an important position in the construction of gas field ground engineering, and its design optimization can significantly improve the economic benefits. At present, some traditional optimization methods are not sufficient for the study of multi-level structures of pipeline networks and the discovery of an effective solution. Therefore, this paper proposes a set of optimization methods for the more complex star–tree gas-gathering pipeline network in discrete spaces.

Based on the literature reviewed, scholars have made significant strides in the design optimization of gas pipeline networks. However, their focus has primarily been on optimizing single-network topologies, neglecting the complexity introduced by multiple topological combinations, particularly the star–tree multi-level configurations prevalent in oil and gas fields. The inclusion of diverse network types significantly escalates the complexity of solving these models. We approach this issue as follows:

- (1) We establish an MINLP model for the optimal design of star–tree networks in discrete space.
- (2) A hierarchical optimization framework is proposed. The first stage focuses on optimizing the design of the star topology pipeline network connecting wells to stations. The second stage involves optimizing the design of the tree topology pipeline network connecting stations to plants.
- (3) An improved genetic algorithm (IGA) is proposed, and the Kruskal algorithm is used to achieve optimal layout solutions for the tree network.

Section 2 delineates the mathematical model. Section 3 introduces a hierarchical optimization framework. Section 4 presents a practical case and examines the optimization result. Finally, Section 5 provides a summary.

2. Mathematical Model

Inside the natural gas-gathering and transportation pipeline network layout, there are two main types of models: one is an MIP model and the other is an MINLP model. In the MIP model, various point-to-point distances are calculated in advance. For the development of gas fields, the scale of the number of wells is very large, and the calculation of the distance in advance is very large; therefore, in the model of this paper, the pipeline length from wells to stations L_{ij} and the pipeline length between stations $L_{jj'}$ are both Euclidean distances, which are nonlinear equations, respectively, $L_{ij} = \left((x_i - x_j)^2 + (y_i - y_j)^2 \right)^{0.5}$ and $L_{jj'} = \left((x_j - x_{j'})^2 + (y_j - y_{j'})^2 \right)^{0.5}$, which can improve the solving efficiency. In addition, the station location decision variable (δ_j), plant location decision variable (δ_k), and pipeline connection decision variable ($\delta_{ij}, \delta_{jj'}$) are binary variables. Therefore, the model in this paper is an MINLP one. The model has good scalability, which would allow for the number of nodes to be increased in the future.

2.1. Objective Function

An MINLP model is proposed. The minimum investment is used as the objective function of the model, as shown in Equation (1).

$$\text{Min } F = F_{PP} + F_{GP} + F_S + F_C \quad (1)$$

where F is the pipeline network investment, CNY. F_{PP} is the production pipeline investment, CNY. F_{GP} is the gathering pipeline investment, CNY. F_S is the station investment, CNY. F_C is the plant investment, CNY.

(1) Production pipeline investment.

The investment in production pipelines is the cost of connecting all wells and stations in the network, as shown in Equation (2).

$$F_{PP} = \sum_{i \in I} \sum_{j \in J} \delta_{ij} c_{ij} L_{ij} \quad (2)$$

where δ_{ij} is the connection decision variable between wells and stations, 0 or 1. c_{ij} is the price of the pipeline between wells and stations, CNY/m. L_{ij} is the pipeline length from wells to stations, m. $L_{ij} = \left((x_i - x_j)^2 + (y_i - y_j)^2 \right)^{0.5}$, (x_i, y_i) is the well coordinate, and (x_j, y_j) is the station coordinate.

(2) Gathering pipeline investment.

The investment in gathering pipelines is the cost of connecting the pipelines between all stations in the network, as shown in Equation (3).

$$F_{GP} = \sum_{j \in J} \sum_{j' \in J} \delta_{jj'} c_{jj'} L_{jj'} \quad (3)$$

where $\delta_{jj'}$ is the connection decision variable between stations, 0 or 1. $c_{jj'}$ is the price of the pipeline between stations, CNY/m. $L_{jj'}$ is the pipeline length between stations, m. $L_{jj'} = \left((x_j - x_{j'})^2 + (y_j - y_{j'})^2 \right)^{0.5}$, (x_j, y_j) , and $(x_{j'}, y_{j'})$ are the stations or plant coordinate.

(3) Station investment.

The station investment is the sum of the investment costs of all stations in the network, as shown in Equation (4).

$$F_S = \sum_{j \in J} \delta_j f_S \quad (4)$$

where δ_j is the station location decision variable, 0 or 1. f_S is the cost of the station, CNY.

(4) Plant investment.

The plant investment is the sum of the investment costs of all plants in the pipeline network, as shown in Equation (5).

$$F_C = \sum_{k \in K} \delta_k f_P \quad (5)$$

where δ_k is the plant location decision variable, 0 or 1. f_P is the cost of plant, CNY.

2.2. Constraints

(1) Uniqueness constraint of well connections.

Each well is exclusively associated with one station, as shown in Equation (6).

$$\sum_{j \in J} \delta_{ij} = 1 \forall i \in I \quad (6)$$

(2) Plant number constraint.

There exists a single plant within the pipeline network, as shown in Equation (7).

$$\sum_{k \in K} \delta_k = 1 \quad (7)$$

(3) Well and station affiliation constraint.

Well i can be assigned to station j when station j exists, as shown in Equation (8).

$$\delta_{ij} \leq \delta_j \quad \forall i \in I, \quad \forall j \in J \quad (8)$$

(4) Transmission radius constraint.

The gas pressure may decrease during transportation due to pipe friction. In order to keep the pressure drop within a reasonable range, the distance between the well and the station should be less than a specific value, known as the transmission radius, as shown in Figure 5.

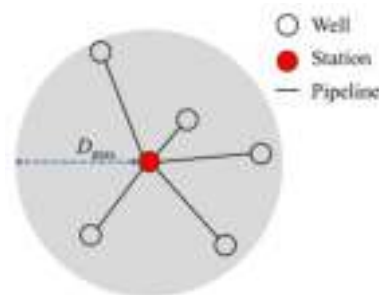


Figure 5. Schematic diagram of the transmission radius.

As such, the transmission radius constraint is calculated as shown in Equation (9).

$$\delta_{ij}L_{ij} \leq D_{\max} \quad \forall i \in I, \forall j \in J \quad (9)$$

where $D_{\max}$ is the maximal allowable pipeline length from the well to the station, m.

(5) Station capacity constraint.

Generally, the gathering radius constraint is not considered when addressing the station capacity constraint, which falls under the capacitated facility location problem (CFLP). In this paper, both the radius constraint and the capacity constraint are considered. When the gathering radius constraint is satisfied but the capacity constraint is not, the pipeline network layout will be redesigned, as shown in Figure 6.

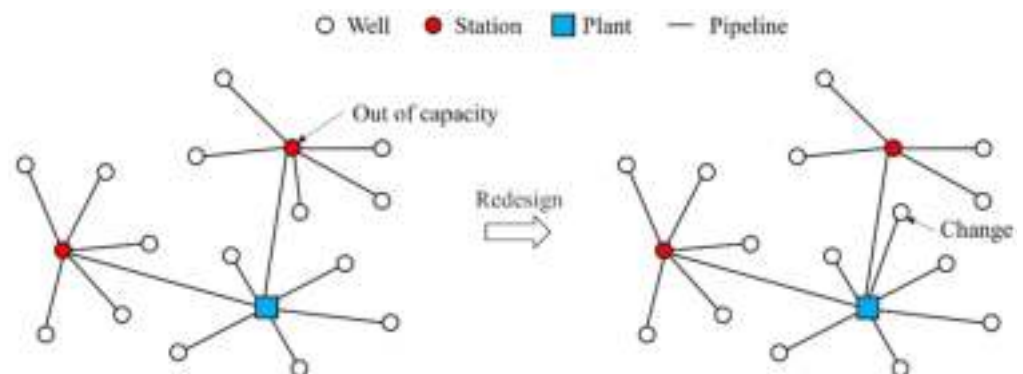


Figure 6. Pipeline network redesigns in violation of capacity constraints.

Thus, the capacity of a station must not exceed that shown in Equation (10).

$$\delta_j q_j \leq q_{\max} \quad \forall j \in J \quad (10)$$

where q_j is the node gas flow of station j , m^3/d . $q_{\max}$ is the capacity of the station, m^3/d .

(6) Tree structure constraint.

The tree connection constraint ensures that the generated topology is the shortest and loop-free, as shown in Equation (11).

$$\sum_{j \in J} \sum_{j' \in J} \delta_{jj'} = 2 \sum_{j \in J} \delta_j \quad (11)$$

(7) Node flow constraint.

The node flow constraint ensures that the total gas volume remains constant, as shown in Equation (12).

$$\sum_{i \in I} q_i - \sum_{k \in K} \delta_k q_k = 0 \quad (12)$$

where q_i is the node gas flow of well i , m^3/d . q_k is the node flow of plant k , m^3/d .

2.3. Decision Variables

The decision variables include the station location decision variable (δ_j), plant location decision variable (δ_k), and pipeline connection decision variable ($\delta_{ij}, \delta_{jj'}$). All these decision variables are binary variables, as shown in Equation (13).

$$\delta_j, \delta_k, \delta_{ij}, \delta_{jj'} = \{0, 1\} \quad i \in I, j \in J, j' \in J, k \in K \quad (13)$$

3. Solving Method

The design optimization of a star–tree pipeline network is recognized as NP-hard, presenting challenges for finding a comprehensive solution [22]. To address this, a hierarchical optimization framework is proposed in this study. The framework consists of two stages: the first stage involves determining well group division and station locations within the star topology network, while the second stage focuses on establishing station connections and plant locations in the tree topology network. Notably, optimizing the star topology network in the first stage serves as the foundational and pivotal component of the entire optimization problem. Its complexity and scale exceed those of the second-stage tree topology network optimization. Therefore, this paper concentrates on enhancing the design optimization and introduces an IGA tailored to address this challenge. Additionally, the Kruskal algorithm, leveraging the minimum spanning tree approach, is employed for the second-stage tree topology pipeline network design optimization.

3.1. Basic Genetic Algorithm

A GA is a heuristic search method that draws inspiration from the principles of natural selection and genetics. It aims to optimize the allocation of facilities by simulating the evolutionary process of biological chromosomes. The basic solving process of GA [23], unfolds as follows:

- (1) Encoding: encode the information on the distribution of wells and stations into the chromosome sequence.
- (2) Generating initial population: a fixed size of chromosomes (individuals) is randomly generated to form the initial population $P(t)$ (in this paper, 200 groups are generated). Here, t denotes the evolutionary generation counter, and T represents the maximum number of evolution cycles set for the algorithm.
- (3) Fitness function: each individual in the population is evaluated based on a fitness function, typically using the total investment value as the criterion for evaluation.
- (4) Selection and inheritance: the initial population is selected and crossed by simulating the process of biological evolution, and the next generation population $P(t + 1)$ is obtained through step 3.
- (5) Termination condition judgment: when $t \leq T$, then $t + 1$ replaces t , and we go to step 2; if $t > T$, we regard the individual with optimal fitness through the process of evolution as the optimal solution and terminate the operation [24].

The above process is also described by the flow diagram shown in Figure 7.

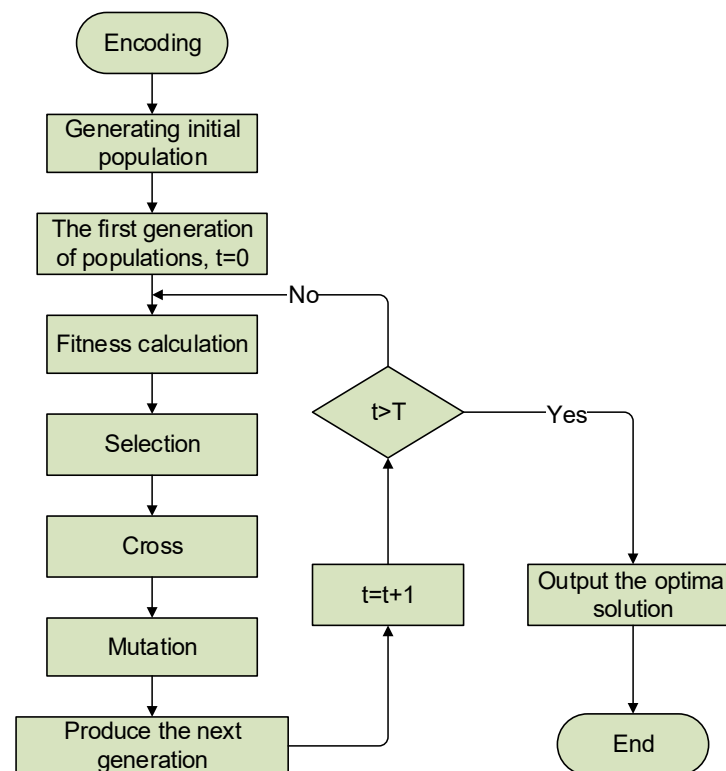


Figure 7. Flow diagram of the basic GA.

3.2. Algorithm Improvement

Considering the discrete nature of the space and the specific structure of gas field pipeline networks, the traditional GA and its associated parameters may not fully align with the mathematical model proposed in this paper. Therefore, it is essential to enhance the basic GA to better suit these characteristics and optimize the pipeline network design effectively. The improvement includes the overall coding of well–station correspondence, the formulation of the fitness function of the network solution, and the redesign of the crossover operator and mutation operator.

(1) Coding design.

Taking into account the characteristics of the constraints, using the connection relationship between wells and stations as a coding basis, real number coding is used to improve the efficiency of the GA. The chromosome-encoding sequence is structured such that it directly impacts the objective function, which in this case is the total investment. Each chromosome's length corresponds to the total number of wells, and within the sequence, the identifier of each well is systematically represented in ascending order. This encoding method ensures that the GA optimally addresses the design constraints and effectively targets the optimization of the gas pipeline network's configuration. Each well's corresponding station number is stochastically generated at every position within the chromosome to denote the linkage between the well and its respective station. In the proposed chromosome coding scheme, each digit in a chromosome = $(C_1, C_2, \dots, C_m)$ represents the number of the station corresponding to each well.

(2) Crossover operator design.

For the GA, the crossover operation method and cross probability are extremely important parameters which directly affect the optimization quality of the algorithm. Using CAP101 data based on OR-LIBRARY (50 wells, 25 stations) [25], the crossover operation method and crossover probability are tested for their influence on the solution, and the best crossover mode and crossover probability are determined.

Each of the crossover techniques was tested 50 times, and their results were averaged for comparison. As the iterative generations increased, the algorithm progressively con-

verged toward the optimal solution. Among the various crossover methods evaluated, the multi-point crossover method demonstrated a superior performance, leading to its selection. Figure 8 illustrates the average test outcomes.

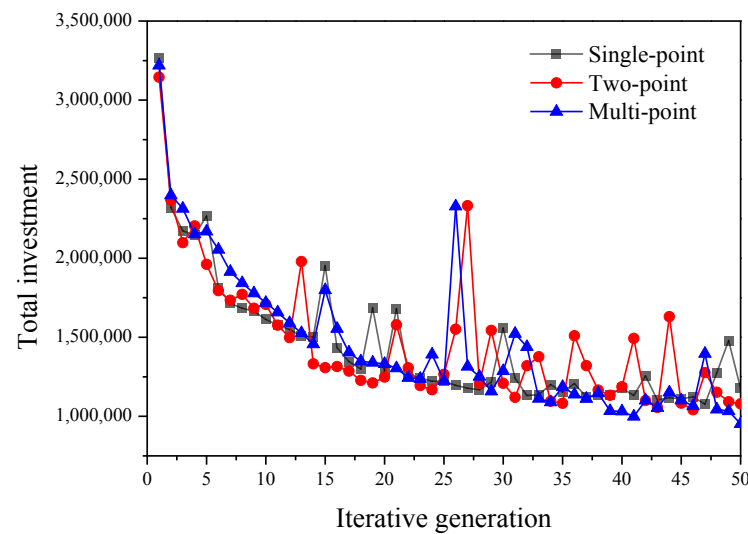


Figure 8. Convergence curve of different crossover operation methods.

The crossover probability varies between 0.5 and 0.9. A comparison of the effects of these different crossover probabilities on the results was conducted by averaging the outcomes from 50 tests. When the crossover probability is between 0.7 and 0.9, the results are similar. Increasing the crossover probability properly can improve the convergence of the solution, so the crossover probability should be kept at a high value in the design. Figure 9 shows the effect of different crossover probabilities on the results in the multi-point crossing method.

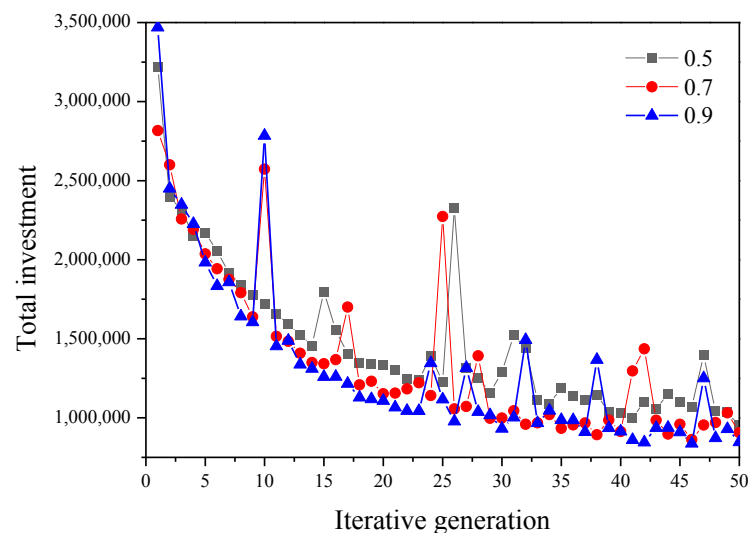


Figure 9. Convergence curve of different crossover probabilities.

(3) Mutation operator design.

The mutation operator is a measure that prevents the GA from falling into the local optimum by simulating the genetic and evolutionary values of the rupture allele [26]. In the process of debugging the mutation operator, a variation function is tested for the GA to

mitigate the risk of falling into local optima in later stages. The mutation operators employ an adaptive approach, as indicated in Formula (14).

$$\begin{cases} MP = \varepsilon \cdot \left(\frac{Fit_{\max} - Fit}{Fit_{\max} - Fit_{\min}} \right) & Fit \geq \frac{Fit_{\max} + Fit_{\min}}{2} \\ MP = \varepsilon & Fit < \frac{Fit_{\max} + Fit_{\min}}{2} \end{cases} \quad (14)$$

where MP is the mutation probability; Fit is the fitness value; $Fit_{\max}$ and $Fit_{\min}$ are the maximum and minimum fitness value of each generation; and ε is the mutation coefficient.

Through testing, it was found that when the mutation coefficient is below 0.003, the iterative stability and convergence of the whole solution process is better, and the solution time is more reasonable; however, if the mutation coefficient has a value of more than 0.1, the determined solution may be far away from the optimal solution, affecting the stability of the iteration and the speed of convergence and other problems. Therefore, the mutation coefficient was chosen to be below 0.003.

The impact of mutation probability on investments is depicted in Table 1. These experiments on CAP101 demonstrate that when the mutation probability is set to a specific value, it ensures population diversity without compromising the convergence of the GA toward the global optimal solution. The main reason for this is that when real number coding is used, each position on each chromosome contains more information than when other encoding methods are used, and when the mutation probability is high, it is easy to directly destroy the dominant individual.

Table 1. Impact of varying mutation probabilities on GA performance.

Problem Size	Total Number of Calculations	ε	Average Optimal Result
50×25	10	0.0001	861,682.74
		0.0002	830,511.15
		0.0003	818,015.08
		0.0004	796,668.16
		0.0005	806,771.78
		0.001	809,352.78
		0.002	855,745.99
		0.003	834,490.13

3.3. IGA Testing

To better address the optimal design of star-tree pipeline networks in discrete spaces, this paper emphasizes enhancements to the coding scheme, crossover operator, and mutation operator of the foundational GA. The solving steps of the IGA are shown in Figure 10.

To validate the effectiveness of the modified genetic algorithm in solving the pipeline network layout design problem, a series of algorithm tests were conducted. All algorithms were implemented in MATLAB 2022 software. The numerical experiments were executed on a computer equipped with an Intel(R) Core (TM) i5-2450M CPU @ 2.50 GHz, 4 GB of RAM, and the Windows 7 operating system.

Two classical algorithms were selected for comparison with the improved GA, namely constriction factor particle swarm optimization (CPSO) and tabu search (TABU). Several problems in the OR-LIBRARY were utilized for algorithmic testing, and the solution outcomes are presented in Table 2. LOCAL is the actual optimal solution. n stands for well number. m stands for station number. For smaller numbers of wells and stations, such as in the case of CAP101, the benefits of using the IGA are not readily apparent. In fact, the IGA may require more computation time compared to other algorithms. However, as the numbers of n (wells) and m (stations) increase, the advantages of the IGA's efficiency in

solving problems become increasingly evident. It tends to converge to the optimal solution more swiftly than other algorithms as n and m grow larger.

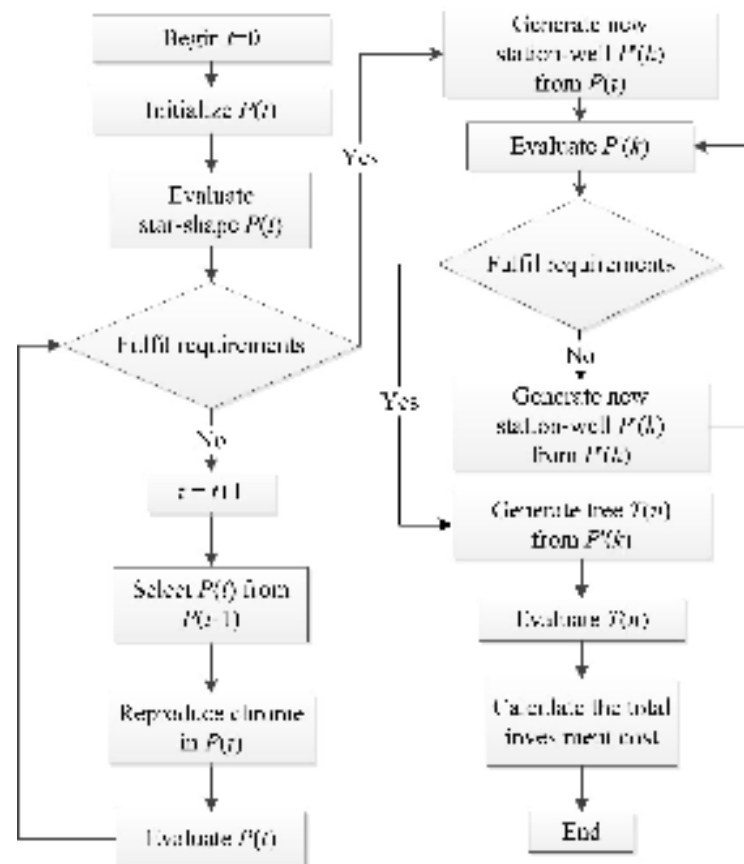


Figure 10. Flow diagram of the IGA.

Table 2. Comparison with other algorithms.

Problem	Size ($n \times m$)	Algorithm	Optimal Solution (Ave.)	Time (s)
CAP101	50×25	LOCAL	796,648.437	-
		TABU	797,508.70	0.19
		CPSO	796,648.44	0.88
		IGA	796,668.16	0.37
CAP103	50×25	LOCAL	893,782.11	-
		TABU	894,008.10	0.24
		CPSO	893,782.11	0.99
		IGA	895,192.66	0.17
CAP104	50×25	LOCAL	928,941.75	-
		TABU	928,941.80	0.15
		CPSO	928,941.75	0.60
		IGA	928,946.48	0.17

Table 2. Cont.

Problem	Size ($n \times m$)	Algorithm	Optimal Solution (Ave.)	Time (s)
CAP131	50×50	LOCAL	793,439.56	-
		TABU	794,299.90	0.20
		CPSO	793,439.56	3.62
		IGA	793,902.48	0.37
CAP133	50×50	LOCAL	893,076.71	-
		TABU	893,782.10	0.30
		CPSO	893,076.71	3.78
		IGA	893,994.01	0.22
CAPB	1000×100	LOCAL	13,198,556.43	-
		TABU	13,214,718.1	3.91
		IGA	13,211,519.82	3.26

3.4. Kruskal Algorithm

When connecting each station using a tree topology, the fundamental task is to address the minimum spanning tree problem. In the design of the minimum spanning tree, commonly used algorithms include the Prim algorithm and Kruskal algorithm, whose kernel is based on the greedy principle. By comparing the two algorithms, it was found that the Kruskal algorithm is always superior to the Prim algorithm in solving time, so we chose to use the Kruskal algorithm as the tree topology network design algorithm. The Kruskal algorithm is a spanning tree algorithm based on weighted edges. In the process of solving, the smallest weighted edge is successively selected and added to the tree set. When the added edge generates a cyclic topology, the weighted edge is removed, until all edges have been selected [27].

The solving process of the Kruskal algorithm is shown in Figure 11. All nodes are divided into the join node set U and the unjoined scatter node set V . At the beginning, $V = \{v_1, v_2, v_3, v_4, v_5\}$ and $U = \{\phi\}$. Firstly, the nodes that form the shortest edge are selected and added to the join node set, that is $U = \{v_4, v_5\}$ and $V = \{v_1, v_2, v_3\}$. Then, in the same way, edges v_1, v_2 , v_1, v_3 , v_4, v_2 , and v_4, v_3 are selected to the join node set. When a cyclic typology is created, the edge that produced it is removed and the solution is complete. Finally, $V = \{\phi\}$ and $U = \{v_1, v_2, v_3, v_4, v_5\}$.

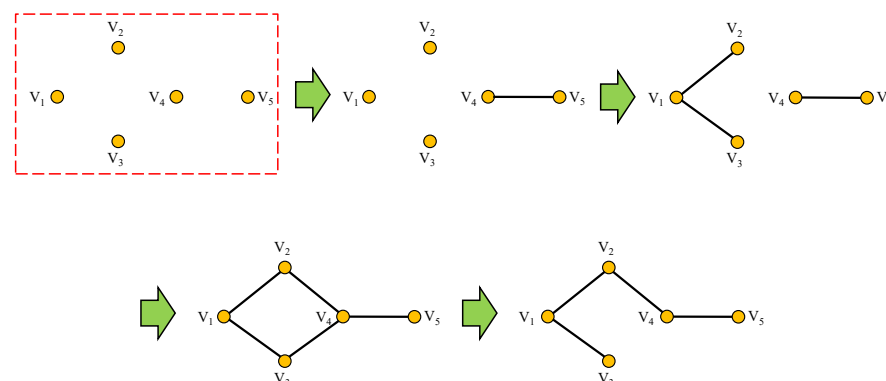


Figure 11. Schematic diagram of Kruskal algorithm solving process.

4. Results and Discussion

For a natural gas field gathering and transmission pipeline network system, the layout design is an NP problem, which is not conducive to calculation. In order to improve the efficiency of the solution, the following four assumptions are made: (1) For the external

conditions of the calculation in the layout design, the standard conditions become the default for calculation (pressure of 0.1 MPa, temperature of 20 °C). (2) The flow rate in the gas pipeline is unstable. In order to solve the problem of convenience, the gas flow in the pipeline is simplified as a stable flow. (3) In the pipeline flow, the temperature of the buried pipeline usually changes with the length of the pipeline, and when the pipeline is too long, the gas will eventually become close to the ambient temperature. In order to facilitate the calculations, and since the gathering pipeline network is usually located in the same area, the temperature of the pipeline is assumed to be constant in this paper. (4) For the direction of the flow of the gas in the network, it is stipulated that each node can only transport the gas from one direction to another.

An actual natural gas pipeline network in China was selected for analysis and search for a solution [28]. The original layout of the pipeline network in the gas field is depicted in Figure 12. This gas field comprises 93 wells, with each producing between 2000 and 5000 m³/d of gas, with wellhead pressures ranging from 0.2 to 1.0 MPa and an average production temperature of 20 °C. The specification of the primary production pipelines between wells and stations was De160 × 9.5 mm, while the specification of the secondary gathering pipelines between stations (and the plant) was De225 × 12.8 mm. Detailed costs are provided in Table 3.

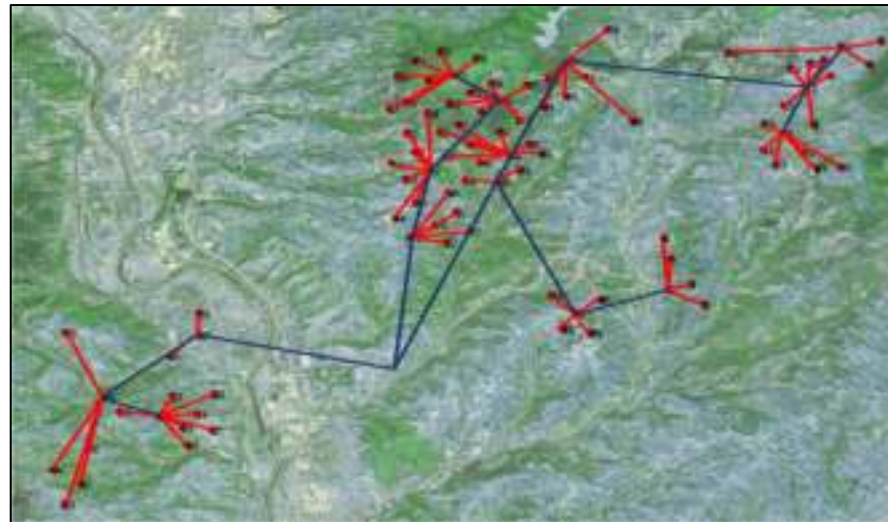


Figure 12. Sketch of the original pipeline network layout.

Table 3. Projected costs of the pipeline network.

Project	Cost
Station	CNY 52.17×10^4
Plant	CNY 1725.93×10^4
Pipeline cost (De160 × 9.5 mm)	CNY/km 15.636×10^4
Pipeline cost (De225 × 12.8 mm)	CNY/km 25.406×10^4

The maximum radius constraint was 1.5 km, and the processing capacity of a single station was 20,000 m³/day. Since the data came from a low-pressure gas field, pressure constraints were not considered in this case. In the original algorithm (GA), the calculation took more than 2 h at a time. After optimizing the algorithm structure (IGA), the calculation time was shortened to 15–30 min, increasing the calculation efficiency by over 400%. The IGA was used to solve the layout of pipeline network.

Utilizing the established hierarchical optimization framework, we derived an optimized layout scheme. The initial star topology layout in the first stage is illustrated in

Figure 13. Figure 14 displays the results of well grouping and station location determination. The final optimized layout is depicted in Figure 15. This network consists of fourteen gas-collecting stations and one treatment plant, which is one station fewer than the original gas field design.



Figure 13. Connection relationship between wells and stations.

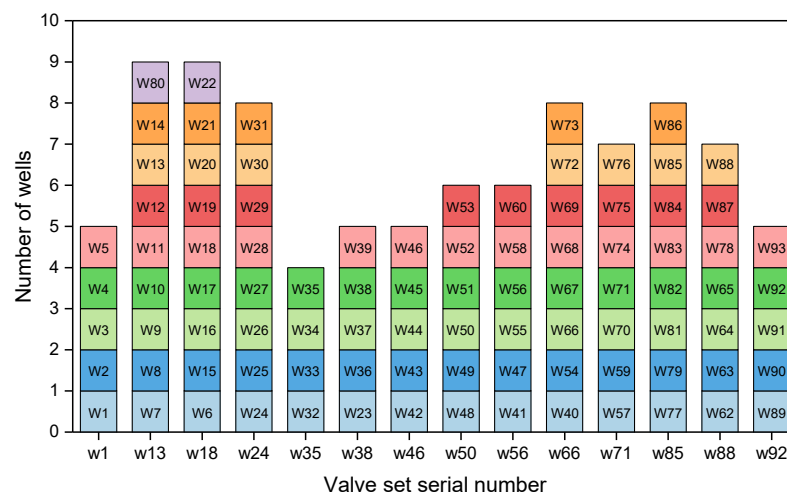


Figure 14. Results for well allocation, station location, and plant location.

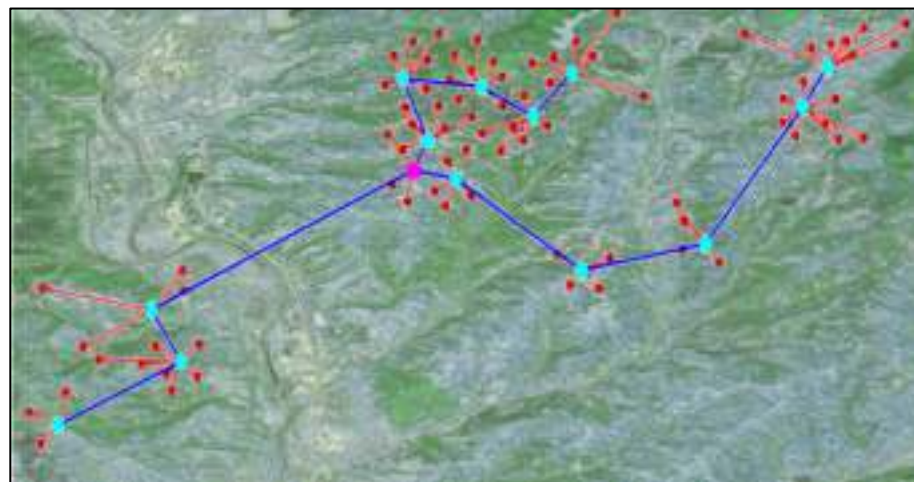


Figure 15. Design result for star-tree pipeline network layout.

Using the simulation software to verify the feasibility of the pipeline network layout structure, and considering that the number of wells in the pipeline network was too large, the number of wells was determined as the sum of the gas volume and the node of the valve manifold for the connection verification. The connection diagram is shown in Figure 16. Corresponding pressure drop calibration is shown in Figure 17. The pressure drop was within the allowable pressure drop range, meeting the requirements of gas transmission.

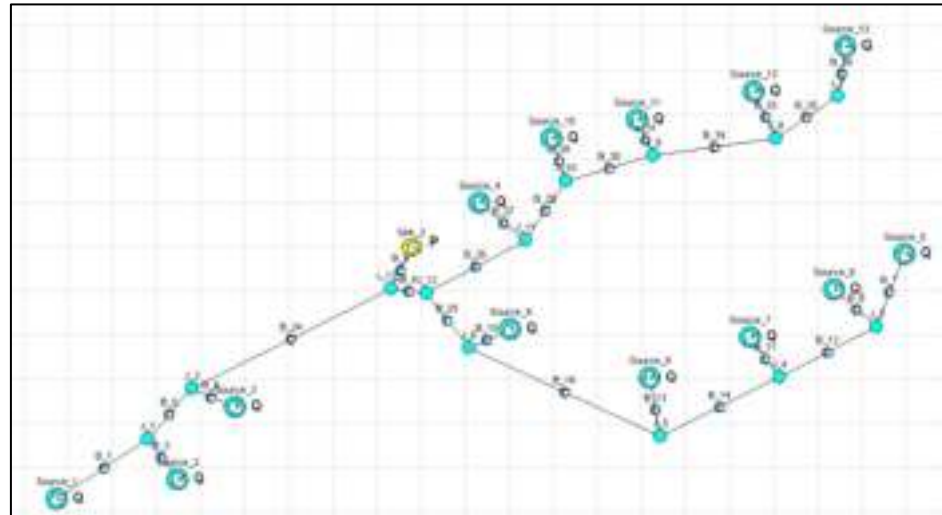


Figure 16. Illustration of pipeline network connection simulation.

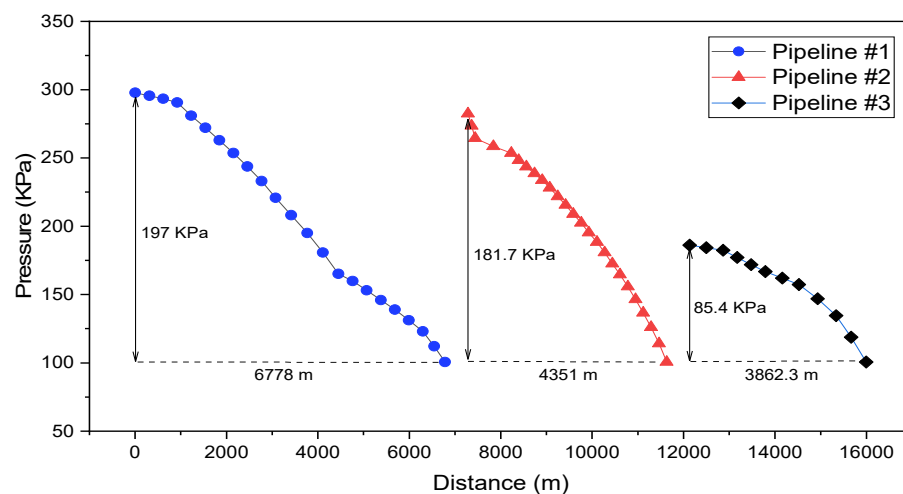


Figure 17. Pipeline network layout pressure drop calibration results.

To highlight the advantages, we also evaluated a star–star pipeline network layout. Figure 18 shows the optimized star–star pipeline network configuration. A comparison of investments between different design schemes is presented in Figure 19. The original layout scheme required a total investment of $\text{CNY } 5650.80 \times 10^4$, while the optimized star–star layout incurred $\text{CNY } 4938.28 \times 10^4$, marking a 12.6% reduction in cost. The optimized star–tree layout scheme further reduced the total investment to $\text{CNY } 4728.28 \times 10^4$, which is 16.3% lower than that of the original scheme. These results demonstrate that both optimized pipeline network layouts achieve cost savings, with the star–tree configuration offering greater advantages. This is attributed to the shorter length of the second-level gathering pipelines between stations and the plant in the star–tree network, resulting in nearly 50% savings in pipeline investment costs and thereby reducing the overall network expenditure.

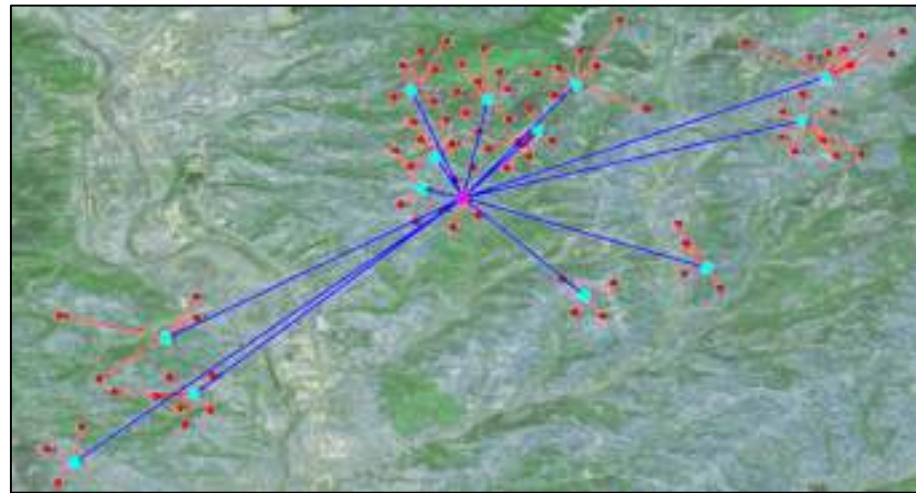


Figure 18. Design result for star-star pipeline network layout.

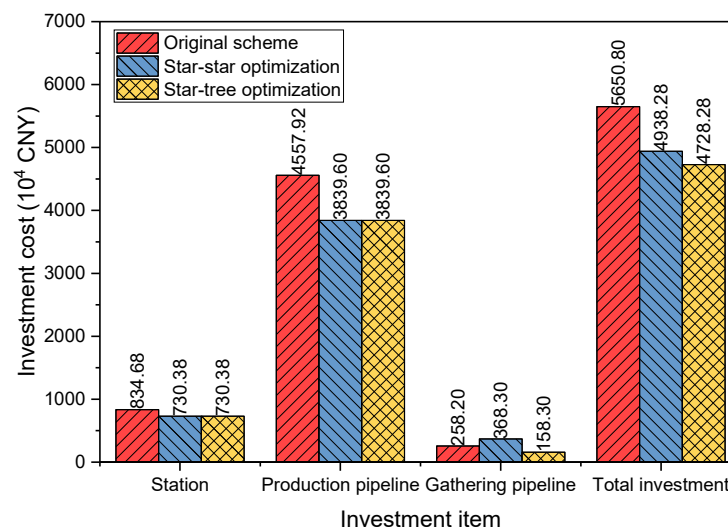


Figure 19. Investment cost comparison.

5. Conclusions

Academic research on optimizing gas pipeline network layouts has predominantly centered on single-level, single-structure networks, with limited attention given to multi-level and multi-type combined gas pipeline networks. Addressing this gap, this paper establishes a design optimization model for a two-stage star-tree gas-gathering pipeline network in discrete spaces. Recognizing the complexity of the problem and the need for computational efficiency, a hierarchical optimization framework is developed alongside an IGA to effectively solve this optimization problem.

Building upon this model, we determined the optimized design of a star-tree pipeline network for a natural gas field in China, resulting in optimal design outcomes. The findings reveal that the optimized network layout scheme achieves a 16% reduction in investment costs compared to the original scheme, amounting to savings of $\text{CNY } 922.52 \times 10^4$. Additionally, we compared this optimized layout with a star-star topology. It was observed that adopting the star-star layout would increase the investment cost by 4% compared to the star-tree layout scheme. This comparison underscores the superior applicability of the star-tree topology for this particular network system.

Although the model established in this paper effectively solves the star-tree pipeline network design optimization problem, it still has the following shortcomings: The model does not consider the hydraulic flow characteristics of natural gas, so it cannot effectively

judge whether the fluid pressure changes in pipeline network meet the technological requirements. Furthermore, the hierarchical optimization strategy employed in this study does not ensure that the solution obtained is the global optimal solution. Therefore, future research should comprehensively incorporate gas operation characteristics into the pipeline network design optimization model. Additionally, adopting a more systematic global optimization method will be essential to effectively address this challenge.

Author Contributions: Validation, Y.Q.; formal analysis, H.C.; investigation, J.H.; software, P.D.; data curation, D.L.; writing—original draft, Y.L. and J.Z.; writing—review and editing, Y.L. and J.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by [the National Natural Science Foundation of China] grant number [51704253].

Data Availability Statement: The data that support the findings of this study are available from the corresponding authors upon reasonable request.

Acknowledgments: The authors gratefully acknowledge the financial support provided by the National Natural Science Foundation of China (51704253) for this project.

Conflicts of Interest: Authors Yu Lin and Yanhua Qiu were employed by the company PetroChina Southwest Oil & Gasfield Company. Author Hao Chen was employed by the company Sichuan Shale Gas Exploration and Development Co., Ltd. Author Dafan Liu was employed by the company CNPC Chuanqing Drilling Engineering Co., Ltd. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Nomenclature

Sets

I	Well set, $i \in I = \{1, 2, \dots, N_I\}$
J	Station candidate position set, $j \in J = \{1, 2, \dots, N_J\}$
K	Plant candidate position set, $k \in K = \{1, 2, \dots, N_K\}$

Parameters

F	Pipeline network investment, CNY
c_{ij}	Price of pipeline between wells and stations, CNY/m
$c_{jj'}$	Price of pipeline between stations, CNY/m
$D_{\max}$	Maximal allowable pipeline length from well to station, m
f_P	Cost of plant, CNY
f_S	Cost of station, CNY
F_C	Plant investment, CNY
F_S	Station investment, CNY
F_{PP}	Production pipeline investment, CNY
F_{GP}	Gathering pipeline investment, CNY
L_{ij}	Pipeline length from wells to stations, m
$L_{jj'}$	Pipeline length between stations, m
q_j	Node gas flow of station j , m^3/d
$q_{\max}$	Capacity of station, m^3/d
q_i	Node gas flow of well i , m^3/d
q_k	Node gas flow of plant k , m^3/d
δ_{ij}	Connection decision variable between wells and stations, 0 or 1
δ_j	Station location decision variable, 0 or 1
$\delta_{jj'}$	Connection decision variable between stations, 0 or 1
δ_k	Plant location decision variable, 0 or 1

References

1. Zhou, J.; Fu, T.; Chen, Y.; Xiao, Y.; Peng, J.; Liang, G. A mixed integer nonlinear programming model for optimal design of natural gas storage surface double-pipe network. *J. Energy Storage* **2021**, *44*, 103379. [\[CrossRef\]](#)
2. Rios-Mercado, R.; Borraz-Sanchez, C. Optimization problems in natural gas transportation systems: A state-of-the-art review. *Appl. Energy* **2015**, *147*, 536–555. [\[CrossRef\]](#)
3. Demissie, A.; Zhu, W.; Belachew, C.T. A multi-objective optimization model for gas pipeline operations. *Comput. Chem. Eng.* **2017**, *100*, 94–103. [\[CrossRef\]](#)
4. Hong, B.; Li, X.; Di, G.; Song, S.; Yu, W.; Chen, S.; Li, Y.; Gong, J. An integrated MILP model for optimal planning of multi-period onshore gas field gathering pipeline system. *Comput. Ind. Eng.* **2020**, *146*, 106479. [\[CrossRef\]](#)
5. Sanaye, S.; Mahmoudimehr, J. Optimal design of a natural gas transmission network layout. *Chem. Eng. Res. Des.* **2013**, *91*, 2465–2476. [\[CrossRef\]](#)
6. Rodriguez, D.A.; Oteiza, P.P.; Brignole, N.B. Simulated Annealing Optimization for Hydrocarbon Pipeline Networks. *Ind. Eng. Chem. Res.* **2013**, *52*, 8579–8588. [\[CrossRef\]](#)
7. Fu, Y.X. Multi-objective parameters optimization design of single-pipe ring-type mixing water oil-gathering pipe network. *Key Eng. Mater.* **2011**, *467*, 1285–1290. [\[CrossRef\]](#)
8. Wang, B.; Yuan, M.; Zhang, H.; Zhao, W.; Liang, Y. An MILP model for optimal design of multi-period natural gas transmission network. *Chem. Eng. Res. Des.* **2018**, *129*, 122–131. [\[CrossRef\]](#)
9. Zhou, J.; Zhou, X.; Liang, G.; Peng, J. An MINLP model for network layout of underground natural gas storage. *J. Intell. Fuzzy Syst.* **2020**, *38*, 4619–4642. [\[CrossRef\]](#)
10. Daskin, M. Network and discrete location models, algorithms and applications. *J. Oper. Res. Soc.* **1997**, *48*, 763–764. [\[CrossRef\]](#)
11. Posta, M.; Ferland, J.A.; Michelon, P. An exact cooperative method for the uncapacitated facility location problem. *Math. Program. Comput.* **2014**, *6*, 199–231. [\[CrossRef\]](#)
12. Balinski, M.L. Integer programming: Methods, uses, computations. *Manag. Sci.* **1965**, *12*, 253–313. [\[CrossRef\]](#)
13. Klincewicz, J.G.; Luss, H.; Rosenberg, E. Optimal and heuristic algorithms for multiproduct uncapacitated facility location. *Eur. J. Oper. Res.* **1986**, *26*, 251–258. [\[CrossRef\]](#)
14. Efrayimson, M.A.; Ray, T.L. A branch-bound algorithm for plant location. *Oper. Res.* **1966**, *14*, 361–368. [\[CrossRef\]](#)
15. Uester, H.; Dilaveroglu, S. Optimization for design and operation of natural gas transmission networks. *Appl. Energy* **2014**, *133*, 56–69. [\[CrossRef\]](#)
16. Zhang, H.; Liang, Y.; Zhang, W.; Wang, B.; Yan, X.; Liao, Q. A unified MILP model for topological structure of production well gathering pipeline network. *J. Pet. Sci. Eng.* **2017**, *152*, 284–293. [\[CrossRef\]](#)
17. Bosikov, I.I.; Klyuev, R.V.; Silaev, I.V.; Stas, G.V. Comprehensive assessment of formalized ventilation difficulty and technological processes in coal mines. *Sustain. Dev. Mt. Territ.* **2023**, *15*, 516–527. [\[CrossRef\]](#)
18. Nguyen, H.; Bui, X.N.; Choi, Y.; Topal, E. Application of artificial intelligence in predicting slope stability in open-pit mines: A case study with a novel imperialist competitive algorithm-based radial basis function neural network. In *Applications of Artificial Intelligence in Mining, Geotechnical and Geoengineering*; Elsevier: Amsterdam, The Netherlands, 2024; pp. 97–111.
19. Rosa, V.R.; Camponogara, E.; Ferreira Filho, V.J.M. Design optimization of oilfield subsea infrastructures with manifold placement and pipeline layout. *Comput. Chem. Eng.* **2018**, *108*, 163–178. [\[CrossRef\]](#)
20. Liu, Y.; Chen, S.; Guan, B.; Xu, P. Layout optimization of large-scale oil-gas gathering system based on combined optimization strategy. *Neurocomputing* **2019**, *332*, 159–183. [\[CrossRef\]](#)
21. Hong, B.; Li, X.; Di, G.; Li, Y.; Liu, X.; Chen, S.; Gong, J. An integrated MILP method for gathering pipeline networks considering hydraulic characteristics. *Chem. Eng. Res. Des.* **2019**, *152*, 320–335. [\[CrossRef\]](#)
22. Montagna, A.F.; Cafaro, D.C.; Grossmann, I.E.; Burch, D.; Shao, Y.; Wu, X.H.; Furman, K. Pipeline network design for gathering unconventional oil and gas production using mathematical optimization. *Optim. Eng.* **2023**, *24*, 539–589. [\[CrossRef\]](#)
23. Slowik, A.; Kwasnicka, H. Evolutionary algorithms and their applications to engineering problems. *Neural Comput. Appl.* **2020**, *32*, 12363–12379. [\[CrossRef\]](#)
24. Tohyama, H.; Ida, K.; Matsueda, J. A genetic algorithm for the uncapacitated facility location problem. *Electr. Commun. Jpn.* **2011**, *94*, 47–54. [\[CrossRef\]](#)
25. Beasley, J.E. OR-Library: Distributing test problems by electronic mail. *J. Oper. Res. Soc.* **1990**, *41*, 1069–1072. [\[CrossRef\]](#)
26. Al-Sultan, K.S.; Hussain, M.F.; Nizami, J.S. A genetic algorithm for the set covering problem. *J. Oper. Res. Soc.* **1996**, *47*, 702–709. [\[CrossRef\]](#)
27. Pettie, S.; Ramachandran, V. An optimal minimum spanning tree algorithm. *J. ACM* **2002**, *49*, 16–34. [\[CrossRef\]](#)
28. Zhou, J.; Peng, J.; Liang, G.; Deng, T. Layout optimization of tree-tree gas pipeline network. *J. Pet. Sci. Eng.* **2019**, *173*, 666–680. [\[CrossRef\]](#)

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.



Article

Fast Minimum Error Entropy for Linear Regression

Qiang Li ¹, Xiao Liao ¹, Wei Cui ¹, Ying Wang ¹, Hui Cao ^{2,*}  and Qingshu Guan ² 

¹ State Grid Information & Telecommunication Group Co., Ltd., Beijing 102209, China; qiangli_sgcc@163.com (Q.L.); xiaoliao_sgcc@163.com (X.L.); weicui_sgcc@163.com (W.C.); yingwang_sgcc@163.com (Y.W.)

² School of Electrical Engineering, Xi'an Jiaotong University, Xi'an 710049, China; qsguan@stu.xjtu.edu.cn

* Correspondence: huicao@mail.xjtu.edu.cn

Abstract: The minimum error entropy (MEE) criterion finds extensive utility across diverse applications, particularly in contexts characterized by non-Gaussian noise. However, its computational demands are notable, and are primarily attributable to the double summation operation involved in calculating the probability density function (PDF) of the error. To address this, our study introduces a novel approach, termed the fast minimum error entropy (FMEE) algorithm, aimed at mitigating computational complexity through the utilization of polynomial expansions of the error PDF. Initially, the PDF approximation of a random variable is derived via the Gram–Charlier expansion. Subsequently, we proceed to ascertain and streamline the entropy of the random variable. Following this, the error entropy inherent to the linear regression model is delineated and expressed as a function of the regression coefficient vector. Lastly, leveraging the gradient descent algorithm, we compute the regression coefficient vector corresponding to the minimum error entropy. Theoretical scrutiny reveals that the time complexity of FMEE stands at $O(n)$, in stark contrast to the $O(n^2)$ complexity associated with MEE. Experimentally, our findings underscore the remarkable efficiency gains afforded by FMEE, with time consumption registering less than 1% of that observed with MEE. Encouragingly, this efficiency leap is achieved without compromising accuracy, as evidenced by negligible differentials observed between the accuracies of FMEE and MEE. Furthermore, comprehensive regression experiments on real-world electric datasets in northwest China demonstrate that our FMEE outperforms baseline methods by a clear margin.

Keywords: minimum error entropy; linear regression; Gram–Charlier expansion; computational complexity



Citation: Li, Q.; Liao, X.; Cui, W.; Wang, Y.; Cao, H.; Guan, Q. Fast Minimum Error Entropy for Linear Regression. *Algorithms* **2024**, *17*, 341. <https://doi.org/10.3390/a17080341>

Academic Editor: Frank Werner

Received: 23 June 2024

Revised: 1 August 2024

Accepted: 3 August 2024

Published: 6 August 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Entropy serves as a pivotal metric for quantifying the inherent uncertainty within a system, finding ubiquitous application across a multitude of domains including dimension reduction [1,2], parameter estimation [3,4], identification [5,6], feedback control [7,8], and abnormal detection [9–11]. The minimum error entropy (MEE) criterion stands out as a notable framework aimed at minimizing the entropy associated with estimation errors, thereby mitigating uncertainty within the estimation model. Particularly pertinent to linear regression estimation quandaries, MEE diverges from conventional least square methodologies by not only considering the variance of prediction errors, but also incorporating higher-order cumulants, rendering it adept at handling non-Gaussian noise distributions [12,13]. Given the prevalence of non-Gaussian noise in real-world scenarios, the efficacy of MEE has been duly validated across various applications, encompassing adaptive filtering [14–16], face recognition [17,18], sparse system identification [19–21], stochastic control systems [22,23], and visible light communication [24].

Furthermore, the convergence properties of MEE have been meticulously examined in the prior literature [25–27]. In-depth analyses delved into the convergence dynamics of fixed-point MEE methodologies [28], while investigations into the interplay between

MEE and Minimum Mean-Squared Error (MMSE) [29], as well as its relationship with maximum correntropy [30], have been extensively explored. Noteworthy extensions to the conventional MEE framework include the proposal of kernel MEE variants [31], elucidations on regularization strategies for MEE implementations [32], and the development of semi-supervised and distributed adaptations [33]. Moreover, within the errors-in-variables (EIV) modeling realm, robustness analyses of MEE methodologies have been expounded upon [34], alongside the proposition of novel methodologies such as the minimum total error entropy approach [35].

The computational demands associated with MEE criterion stem from its time complexity, which scales quadratically as $O(n^2)$. Central to MEE's computational load is the necessity for a double summation operation during the calculation of the gradient pertaining to the error probability density function (PDF), particularly when employing Parzen windowing [36] and certain types of Kernel functions. Moreover, the selection of an inappropriate Kernel function or its parameters can detrimentally impact the accuracy of the resulting PDF. To solve the problem, efforts have been made to optimize the gradient descent MEE approach, as evidenced by the normalization of the step size in [37], leveraging the power of the input entropy. Additionally, innovative methodologies such as the utilization of a quantization operator, as proposed in [38], have been instrumental in mitigating computational complexity. By mapping the error onto a series of real-valued code words, this approach effectively transforms the double summation operation into a singular form, consequently reducing the algorithm's computational overhead.

This paper presents a novel approach, the fast minimum error entropy estimation (FMEE) algorithm, tailored specifically for linear regression tasks, leveraging the polynomial expansion technique applied to the error Probability Density Function (PDF). In contrast to the traditional minimum error entropy (MEE) methodology employing Parzen windowing for error PDF derivation, FMEE adopts the Gram–Charlier expansion method to approximate the error PDF, resulting in a notable reduction in time complexity from $O(n^2)$ to $O(n)$. Notably, FMEE obviates the need for Kernel functions or their associated parameters. The proposed algorithm entails several key steps: firstly, the derivation of the PDF approximation for a random variable via the Gram–Charlier expansion, followed by the simplification of the entropy of the random variable facilitated by the orthogonality of the Hermite polynomials. Subsequently, the error of the linear regression model is ascertained, with the error entropy expressed as a function of the regression coefficient vector. Ultimately, the gradient descent technique is employed to deduce the regression coefficient vector corresponding to the minimum error entropy. Experimental validation underscores the efficacy of FMEE, revealing a remarkable reduction in time consumption, with a disparity of less than 1% compared to that of the MEE approach for identical problem settings. Importantly, minimal discrepancies are observed between the accuracy levels of FMEE and MEE methodologies.

The rest of the paper is organized as follows: Section 2 introduces the related algorithms, FMEE is proposed in detail in Section 3, Section 4 contains the experiments, and the conclusions are in Section 5.

2. The Related Algorithms

2.1. MEE

MEE minimizes the information contained in the error and maximizes the information captured by the estimated model. Usually, the second-order Renyi entropy [39] is used to measure the entropy contained in the error of the model.

$$H(E) = -\log E p(p_E) = -\log \int p_E^2(e) de, \quad (1)$$

where E is the error random variable of MEE, and $H(E)$ is the entropy of E and $p_E(e)$ donates the PDF of E . In this paper, $Ep(\cdot)$ donates the expectation in order to avoid misunderstanding. Then, the PDF of E can be estimated by Parzen windowing:

$$\hat{p}_E(e) = \frac{1}{n} \sum_{i=1}^n K_h(e - e_i), \quad (2)$$

where n is the number of the observed data, K is the kernel function, and h is the bandwidth. Usually, the Gaussian kernel is chosen, and $K_h(z) = \frac{1}{\sqrt{2\pi}h} \exp(-\frac{z^2}{2h^2})$. Then, the empirical information of error $\hat{H}(f)$ can be calculated as:

$$\hat{H}(f) = -\log \frac{1}{n^2} \sum_{i=1}^n \sum_{j=1}^n K_h(e_i - e_j). \quad (3)$$

As the logarithmic function is monotone increasing, it can be removed without any influence on the minimizing process. Then, the transformed empirical error information can be obtained as Equation (4):

$$R(e) = -\frac{1}{n^2} \sum_{i=1}^n \sum_{j=1}^n K_h(e_i - e_j). \quad (4)$$

For the linear regression, $y = \mathbf{w}^T \mathbf{x} + e$, the target of the model is to estimate $\mathbf{w}$ from the limited samples. As $e_i = y_i - \mathbf{w}^T \mathbf{x}_i$, the corresponding empirical error information is calculated as Equation (5):

$$R(\mathbf{w}) = -\frac{1}{n^2} \sum_{i=1}^n \sum_{j=1}^n K_h((y_i - \mathbf{w}^T \mathbf{x}_i) - (y_j - \mathbf{w}^T \mathbf{x}_j)). \quad (5)$$

Then, the MEE estimator $\hat{\mathbf{w}}$ can be obtained by minimizing $R(\mathbf{w})$ with gradient descent. The time complexity of MEE is $O(n^2)$ due to the double summation operation, as shown in Equation (5). As the time consumption of MEE is quadratic to the number of the observed data, the running time of MEE would increase dramatically with the increase in the amount of the observed data.

2.2. The Hermite Polynomials

The Hermite polynomials [40] are a classical orthogonal polynomial sequence. They are defined by the derivatives of the standardized Gaussian PDF $\varphi(\xi)$ as Equation (6):

$$\frac{\partial^i \varphi(\xi)}{\partial \xi^i} = (-1)^i H_i(\xi) \varphi(\xi), \quad (6)$$

where $H(i)$ donates the i th-order Hermite polynomial.

The Hermite polynomials forms an orthogonal system, as in Equation (7):

$$\int \varphi(\xi) H_i(\xi) H_j(\xi) d\xi = \begin{cases} i!, & \text{if } i = j \\ 0, & \text{if } i \neq j \end{cases}, \quad (7)$$

This paper uses the first five Hermite polynomials, shown in Equation (8):

$$\begin{aligned} H_0(x) &= 1, \\ H_1(x) &= x, \\ H_2(x) &= x^2 - 1, \\ H_3(x) &= x^3 - 3x, \\ H_4(x) &= x^4 - 6x^2 + 3. \end{aligned} \quad (8)$$

2.3. The Gram–Charlier Expansion

The Gram–Charlier expansion obtains the approximation of the PDF of a random variable, and there is no significant loss of accuracy with this expansion, which is demonstrated in [41]. Under the assumption that the random variable is near the normal distribution with the same mean and variance as itself, the Gram–Charlier expansion of the PDF of x can be expressed as:

$$p_x(\xi) \approx \hat{p}_x(\xi) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(\xi - \mu)^2}{2\sigma^2}\right] * \left[1 + \frac{\kappa_3(x)}{3!\sigma^3} H_3\left(\frac{\xi - \mu}{\sigma}\right) + \frac{\kappa_4(x)}{4!\sigma^4} H_4\left(\frac{\xi - \mu}{\sigma}\right)\right], \quad (9)$$

where μ and σ are the mean and standard deviation of x . ξ is the Gaussian random variable whose mean and variable are the same as those of x . $\kappa_3(x)$ and $\kappa_4(x)$ are the third-order and fourth-order cumulant of x , which can also be called skewness and kurtosis. H_3 and H_4 are the third-order and fourth-order Hermite polynomials.

3. Methodology

This section derives the relation between the differential entropy and the Gram–Charlier expansion of the PDF of the error for linear regression. Moreover, the assumption that the PDF of the regression error is near the Gaussian distribution $\varphi(\xi)$ that has the same mean and variance as the error is implemented.

The differential entropy [42] of a random variable X is shown in Equation (10):

$$H(x) = - \int p(x) \log p(x) dx. \quad (10)$$

Substitute Equation (9) into Equation (10); the result can be derived in Equation (11):

$$\begin{aligned} H(x) &\approx - \int \hat{p}_x(\xi) \log \hat{p}_x(\xi) d\xi \\ &= - \int \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(\xi - \mu)^2}{2\sigma^2}\right] * \left[1 + \frac{\kappa_3(x)}{3!\sigma^3} H_3\left(\frac{\xi - \mu}{\sigma}\right) + \frac{\kappa_4(x)}{4!\sigma^4} H_4\left(\frac{\xi - \mu}{\sigma}\right)\right] * \\ &\quad \log \left[\frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(\xi - \mu)^2}{2\sigma^2}\right] * \left[1 + \frac{\kappa_3(x)}{3!\sigma^3} H_3\left(\frac{\xi - \mu}{\sigma}\right) + \frac{\kappa_4(x)}{4!\sigma^4} H_4\left(\frac{\xi - \mu}{\sigma}\right)\right] \right] d\xi. \end{aligned} \quad (11)$$

This integral is rather difficult to compute. However, notice that if the PDF of x is near the normal density as assumed, it can be inferred that $\kappa_3(x)$ and $\kappa_4(x)$ are very small. Then, the approximation in Equation (12) can be used:

$$\log(1 + \epsilon) \approx \epsilon - \frac{\epsilon^2}{2}, \quad (12)$$

where ϵ denotes the infinitesimal.

Then, Equation (11) can be transformed into Equation (13):

$$\begin{aligned}
H(x) = & - \int \varphi(\xi) * \\
& \left[1 + \frac{\kappa_3(x)}{3!\sigma^3} H_3\left(\frac{\xi - \mu}{\sigma}\right) + \frac{\kappa_4(x)}{4!\sigma^4} H_4\left(\frac{\xi - \mu}{\sigma}\right) \right] * \\
& \left[\log \varphi(\xi) + \frac{\kappa_3(x)}{3!\sigma^3} H_3\left(\frac{\xi - \mu}{\sigma}\right) + \right. \\
& \frac{\kappa_4(x)}{4!\sigma^4} H_4\left(\frac{\xi - \mu}{\sigma}\right) - \frac{1}{2} \left(\frac{\kappa_3(x)}{3!\sigma^3} H_3\left(\frac{\xi - \mu}{\sigma}\right) + \right. \\
& \left. \left. \frac{\kappa_4(x)}{4!\sigma^4} H_4\left(\frac{\xi - \mu}{\sigma}\right) \right)^2 \right] d\xi.
\end{aligned} \tag{13}$$

As H_i forms an orthogonal system, Equation (13) can be simplified as Equation (14),

$$H(x) = \frac{1}{2} \log 2\pi e \sigma^2 - \frac{\kappa_3^2(x)}{2 * 3!\sigma^6} - \frac{\kappa_4^2(x)}{2 * 4!\sigma^8}. \tag{14}$$

The details of the simplification process of Equation (14) are illustrated as follows. This part is to derive Equation (14). First, it is needed to show that

$$\int \varphi(\xi) \left(\frac{\xi - \mu}{\sigma}\right)^2 H_3\left(\frac{\xi - \mu}{\sigma}\right) d\xi = 0 \tag{15}$$

and

$$\int \varphi(\xi) \left(\frac{\xi - \mu}{\sigma}\right)^2 H_4\left(\frac{\xi - \mu}{\sigma}\right) d\xi = 0 \tag{16}$$

For Equation (15), suppose that $t = (\xi - \mu)/\sigma$,

$$\begin{aligned}
& \int \varphi(\xi) \left(\frac{\xi - \mu}{\sigma}\right)^2 H_3\left(\frac{\xi - \mu}{\sigma}\right) d\xi \\
&= \int \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(\xi - \mu)^2}{2\sigma^2}\right] \left[\left(\frac{\xi - \mu}{\sigma}\right)^3 - 3\left(\frac{\xi - \mu}{\sigma}\right)\right] d\xi \\
&= \int \frac{1}{\sqrt{2\pi}} t^2 \exp\left(-\frac{t^2}{2}\right) (t^3 - 3t) dt
\end{aligned} \tag{17}$$

As $\exp(-\frac{t^2}{2})$ is an even function and $t^5 - 3t^3$ is an odd function, then

$$\int \frac{1}{\sqrt{2\pi}} t^2 \exp\left(-\frac{t^2}{2}\right) (t^3 - 3t) dt = 0 \tag{18}$$

and Equation (15) holds.

For Equation (16), suppose that $t = (\xi - \mu)/\sigma$,

$$\begin{aligned}
& \int \varphi(\xi) \left(\frac{\xi - \mu}{\sigma}\right)^2 H_4\left(\frac{\xi - \mu}{\sigma}\right) d\xi \\
&= \int \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(\xi - \mu)^2}{2\sigma^2}\right] * \\
& \quad \left[\left(\frac{\xi - \mu}{\sigma}\right)^6 - 6\left(\frac{\xi - \mu}{\sigma}\right)^4 + 3\left(\frac{\xi - \mu}{\sigma}\right)^2\right] d\xi \\
&= \int \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{t^2}{2}\right) (t^6 - 6t^4 + 3t^2) dt \\
&= 3 \int \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{t^2}{2}\right) t^2 dt - 6 \int \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{t^2}{2}\right) t^4 dt \\
& \quad + \int \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{t^2}{2}\right) t^6 dt
\end{aligned} \tag{19}$$

The first term in Equation (19) can be calculated as Equation (20),

$$\begin{aligned}
 & \int \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{t^2}{2}\right) t^2 dt \\
 &= \int \frac{1}{\sqrt{2\pi}} (-t) d\left(\exp\left(-\frac{t^2}{2}\right)\right) \\
 &= \frac{1}{\sqrt{2\pi}} (-t) \left(\exp\left(-\frac{t^2}{2}\right)\right) \Big|_{-\infty}^{+\infty} + \\
 & \quad \int \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{t^2}{2}\right) dt \\
 &= 1
 \end{aligned} \tag{20}$$

The second term in Equation (19) can be calculated as Equation (21),

$$\begin{aligned}
 & \int \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{t^2}{2}\right) t^4 dt \\
 &= \int \frac{1}{\sqrt{2\pi}} (-t^3) d\left(\exp\left(-\frac{t^2}{2}\right)\right) \\
 &= \frac{1}{\sqrt{2\pi}} (-t^3) \left(\exp\left(-\frac{t^2}{2}\right)\right) \Big|_{-\infty}^{+\infty} + \\
 & \quad 3 \int \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{t^2}{2}\right) t^2 dt \\
 &= 3
 \end{aligned} \tag{21}$$

The last term in Equation (19) can be calculated as Equation (22),

$$\begin{aligned}
 & \int \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{t^2}{2}\right) t^6 dt \\
 &= \int \frac{1}{\sqrt{2\pi}} (-t^5) d\left(\exp\left(-\frac{t^2}{2}\right)\right) \\
 &= \frac{1}{\sqrt{2\pi}} (-t^5) \left(\exp\left(-\frac{t^2}{2}\right)\right) \Big|_{-\infty}^{+\infty} + \\
 & \quad 5 \int \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{t^2}{2}\right) t^4 dt \\
 &= 15
 \end{aligned} \tag{22}$$

According to Equations (20)–(22),

$$\int \varphi(\xi) \left(\frac{\xi - \mu}{\sigma}\right)^2 H_4\left(\frac{\xi - \mu}{\sigma}\right) d\xi = 15 - 6 * 3 + 3 = 0 \tag{23}$$

and Equation (16) holds.

Then, for Equation (13),

$$\begin{aligned}
 H(x) = & - \int \varphi(\xi) \log \varphi(\xi) d\xi - \\
 & \int \varphi(\xi) \left[\frac{\kappa_3(x)}{3!\sigma^3} H_3\left(\frac{\xi - \mu}{\sigma}\right) + \frac{\kappa_4(x)}{4!\sigma^4} H_4\left(\frac{\xi - \mu}{\sigma}\right) \right] d\xi + \\
 & \int \frac{1}{2} \varphi(\xi) \left[\frac{\kappa_3(x)}{3!\sigma^3} H_3\left(\frac{\xi - \mu}{\sigma}\right) + \frac{\kappa_4(x)}{4!\sigma^4} H_4\left(\frac{\xi - \mu}{\sigma}\right) \right]^2 d\xi - \\
 & \int \varphi(\xi) \log \varphi(\xi) \left[\frac{\kappa_3(x)}{3!\sigma^3} H_3\left(\frac{\xi - \mu}{\sigma}\right) + \right. \\
 & \left. \frac{\kappa_4(x)}{4!\sigma^4} H_4\left(\frac{\xi - \mu}{\sigma}\right) \right] d\xi \\
 & - \int \varphi(\xi) \left[\frac{\kappa_3(x)}{3!\sigma^3} H_3\left(\frac{\xi - \mu}{\sigma}\right) + \right. \\
 & \left. \frac{\kappa_4(x)}{4!\sigma^4} H_4\left(\frac{\xi - \mu}{\sigma}\right) \right]^2 d\xi + \\
 & \int \frac{1}{2} \varphi(\xi) \left[\frac{\kappa_3(x)}{3!\sigma^3} H_3\left(\frac{\xi - \mu}{\sigma}\right) + \frac{\kappa_4(x)}{4!\sigma^4} H_4\left(\frac{\xi - \mu}{\sigma}\right) \right]^3 d\xi
 \end{aligned} \tag{24}$$

Notice that $H_0 = 1$, then the second term of Equation (24) is 0. As it is assumed that x is close to a normal random variable, $\kappa_3(x)$ and $\kappa_4(x)$ are very small. Then, the last term of Equation (24) can be omitted as 0, as it involves third-order monomials, which are infinitely smaller than the terms only containing second-order monomials. Then, for Equation (24), take Equations (15) and (16) into consideration, and notice again that $H_0 = 1$,

$$\begin{aligned}
 H(x) = & H(\xi) - \int \frac{1}{2} \varphi(\xi) \left[\frac{\kappa_3(x)}{3!\sigma^3} H_3\left(\frac{\xi - \mu}{\sigma}\right) \right]^2 + \\
 & \left[\frac{\kappa_4(x)}{4!\sigma^4} H_4\left(\frac{\xi - \mu}{\sigma}\right) \right]^2 + \\
 & 2 \frac{\kappa_3(x)}{3!\sigma^3} H_3\left(\frac{\xi - \mu}{\sigma}\right) \frac{\kappa_4(x)}{4!\sigma^4} H_4\left(\frac{\xi - \mu}{\sigma}\right) d\xi \\
 & - \int \varphi(\xi) \left[\log \frac{1}{\sqrt{2\pi}\sigma} + \log \exp\left(-\frac{\xi^2}{2}\right) \right] * \\
 & \left[\frac{\kappa_3(x)}{3!\sigma^3} H_3\left(\frac{\xi - \mu}{\sigma}\right) + \frac{\kappa_4(x)}{4!\sigma^4} H_4\left(\frac{\xi - \mu}{\sigma}\right) \right] d\xi \\
 = & H(\xi) - \frac{\kappa_3^2(x)}{18\sigma^6} - \frac{\kappa_4^2(x)}{32\sigma^8} - \\
 & \log \frac{1}{\sqrt{2\pi}\sigma} \int \varphi(\xi) \left[\frac{\kappa_3(x)}{3!\sigma^3} H_3\left(\frac{\xi - \mu}{\sigma}\right) + \right. \\
 & \left. \frac{\kappa_4(x)}{4!\sigma^4} H_4\left(\frac{\xi - \mu}{\sigma}\right) \right] d\xi + \frac{1}{2 \ln 2} \int \varphi(\xi) \xi^2 * \\
 & \left[\frac{\kappa_3(x)}{3!\sigma^3} H_3\left(\frac{\xi - \mu}{\sigma}\right) + \frac{\kappa_4(x)}{4!\sigma^4} H_4\left(\frac{\xi - \mu}{\sigma}\right) \right] d\xi \\
 = & \frac{1}{2} \log 2\pi e \sigma^2 - \frac{\kappa_3^2(x)}{18\sigma^6} - \frac{\kappa_4^2(x)}{32\sigma^8}
 \end{aligned} \tag{25}$$

So, Equation (14) holds.

For the linear regression, $e_i = y_i - w^T x_i$, the standard deviation of the error e_w is in Equation (26):

$$\sigma^2(e_w) = Ep(E_w^2) - [Ep(E_w)]^2 = Ep(E_w^2) - (\bar{y} - w^T \bar{x})^2, \tag{26}$$

where $\bar{y}$ is the mean of y and $\bar{x}$ denotes the mean of x .

To simplify the calculation of $\kappa_4(e_w)$, let e_w become a random variable with 0 mean:

$$e_{w_i} = y_i - \mathbf{w}^T \mathbf{x}_i - \bar{y} + \mathbf{w}^T \bar{\mathbf{x}}. \quad (27)$$

Notice that the transformation in Equation (27) does not change the standard deviation and the entropy of e_w as $\sigma^2(e+c) = \sigma^2(e)$ and $H(e+c) = H(e)$ when c is a constant number. In the rest of the paper, to keep clarity, $\sigma^2(\cdot)$ is replaced with $\text{var}(\cdot)$.

Then, for linear regression, its entropy of the error can be expressed as Equation (28):

$$H(e_w) = \frac{1}{2} \log 2\pi e + \frac{1}{2} \log \text{var}(e_w) - \frac{\kappa_3^2(e_w)}{2 * 3! \text{var}^3(e_w)} - \frac{\kappa_4^2(e_w)}{2 * 4! \text{var}^4(e_w)}, \quad (28)$$

where

$$\kappa_3(e_w) = \text{Ep}[(y - \mathbf{w}^T \mathbf{x} - \bar{y} + \mathbf{w}^T \bar{\mathbf{x}})^3], \quad (29)$$

$$\kappa_4(e_w) = \text{Ep}[(y - \mathbf{w}^T \mathbf{x} - \bar{y} + \mathbf{w}^T \bar{\mathbf{x}})^4] - 3. \quad (30)$$

To minimize $H(e_w)$, the derivative of $H(e_w)$ with respect to w is calculated in Equation (31):

$$\begin{aligned} \frac{\partial H(e_w)}{\partial \mathbf{w}} &= \frac{1}{2 \text{var}(e_w) \ln 2} \frac{\partial \text{var}(e_w)}{\partial \mathbf{w}} - \frac{1}{2 * 3! \text{var}^6(e_w)} * \\ &\quad [2 \text{var}^3(e_w) \kappa_3(e_w) \frac{\partial \kappa_3(e_w)}{\partial \mathbf{w}} - \\ &\quad 3 \kappa_3^2(e_w) \text{var}^2(e_w) \frac{\partial \text{var}(e_w)}{\partial \mathbf{w}}] \\ &\quad - \frac{1}{2 * 4! \text{var}^8(e_w)} [2 \text{var}^4(e_w) \kappa_4(e_w) \frac{\partial \kappa_4(e_w)}{\partial \mathbf{w}} - \\ &\quad 4 \kappa_4^2(e_w) \text{var}^3(e_w) \frac{\partial \text{var}(e_w)}{\partial \mathbf{w}}], \end{aligned} \quad (31)$$

where

$$\frac{\partial \text{var}(e_w)}{\partial \mathbf{w}} = -2 \text{Ep}[(y - \mathbf{w}^T \mathbf{x}) \mathbf{x}] + 2(\bar{y} - \mathbf{w}^T \bar{\mathbf{x}}) \bar{\mathbf{x}}, \quad (32)$$

$$\frac{\partial \kappa_3(e_w)}{\partial \mathbf{w}} = 3 \text{Ep}[(y - \mathbf{w}^T \mathbf{x} - \bar{y} + \mathbf{w}^T \bar{\mathbf{x}})^2 (-\mathbf{x} + \bar{\mathbf{x}})], \quad (33)$$

$$\frac{\partial \kappa_4(e_w)}{\partial \mathbf{w}} = 4 \text{Ep}[(y - \mathbf{w}^T \mathbf{x} - \bar{y} + \mathbf{w}^T \bar{\mathbf{x}})^3 (-\mathbf{x} + \bar{\mathbf{x}})]. \quad (34)$$

Then, the optimal w for the minimum of $H(e_w)$ can be obtained by the iteration scheme in Equation (35) with gradient descent:

$$\hat{\mathbf{w}}_{k+1} = \hat{\mathbf{w}}_k - \alpha \frac{\partial H(e_{w_k})}{\partial \hat{\mathbf{w}}_k}, \quad (35)$$

where α is the step size determined by Armijo condition [43].

To compute $\frac{\partial H(e_{w_k})}{\partial \mathbf{w}_k}$ for FMEE in every iteration, $\text{var}(e_{w_k})$, $\kappa_3(e_{w_k})$, $\kappa_4(e_{w_k})$, $\frac{\partial \text{var}(e_{w_k})}{\partial \mathbf{w}_k}$, and $\frac{\partial \kappa_4(e_{w_k})}{\partial \mathbf{w}_k}$ are needed. From Equations (26) and (29)–(34), the time complexity of $\text{var}(e_{w_k})$, $\kappa_3(e_{w_k})$, $\kappa_4(e_{w_k})$, $\frac{\partial \text{var}(e_{w_k})}{\partial \mathbf{w}_k}$, $\frac{\partial \kappa_3(e_{w_k})}{\partial \mathbf{w}_k}$, and $\frac{\partial \kappa_4(e_{w_k})}{\partial \mathbf{w}_k}$ is $O(n)$, respectively. Then, the computational complexity of FMEE is $O(n)$. As the time complexity of MEE is $O(n^2)$ due to the double summation operation, FMEE runs faster than MEE does.

4. Experiments

In this section, a comprehensive array of experiments is undertaken, encompassing both numerical simulations and real-world scenarios. The numerical simulations serve to validate the efficacy of the FMEE approach and to scrutinize the time efficiency of both FMEE and MEE methodologies. Concurrently, practical experiments are conducted aimed at forecasting power outages within a city located in northwest China.

Regarding the numerical simulations, we consider a scenario where $x \in \mathbb{R}^{10}$, with the model defined as $y = w *^T x + e$, where $w * = [1, -1, 1, -1, 1, -1, 1, -1, 1, -1]^T$ and x follows a normal distribution $\mathcal{N}(0, I_{10})$. Two distinct types of noise are examined in our experiments. Firstly, Gaussian noise $e \sim \mathcal{N}(0, 1)$ is employed, while the second type comprises generalized Gaussian noise characterized by a probability density function $f(e) \propto \exp(-\alpha|e|)^{0.3}$. This heavy-tailed distribution, utilized herein, aligns with previous works, such as references [27,35]. Although FMEE is derived under the assumption of near-Gaussian noise, it is anticipated that FMEE remains effective in the presence of non-Gaussian, heavy-tailed noises. Here, α denotes a constant adjusted to maintain a variance of 1 for e . For MEE, the step size is set to $\sqrt{0.005\pi}$, and the scale parameter for the Gaussian kernel is 10, mirroring the settings adopted in Reference [27]. Across the experiments, sample sizes range from 100 to 500, with each combination of sample size and noise type repeated 100 times. The ensuing experimental results are detailed below.

Table 1 provides a comparative analysis of the time consumption between MEE and FMEE in the presence of Gaussian noise. FMEE demonstrates a notable acceleration in computational efficiency compared to MEE. Specifically, while the computational time of MEE exhibits quadratic growth relative to the sample size, FMEE's computational overhead scales linearly with the number of samples. Notably, FMEE achieves remarkable speed, processing 500 objects in approximately 0.1 s, thereby positioning it as a highly promising candidate for time-sensitive applications.

Table 1. The running time of MEE and FMEE for Gaussian noise.

Sample Size	Average Running Time		Optimal Running Time	
	MEE	FMEE	MEE	FMEE
100	103.89	0.0401	77.77	0.0156
200	384.31	0.0580	310.05	0.0312
300	801.52	0.0769	654.21	0.0468
400	1430.61	0.1000	1254.12	0.0468
500	2306.24	0.1104	2072.50	0.0624

Table 2 compares the mean squared error between MEE and FMEE in the context of Gaussian noise. Overall, MEE exhibits superior performance compared to FMEE in terms of MSE. Particularly noteworthy is MEE's substantial advantage over FMEE, especially evident when the sample size is 100. However, as the sample size increases, the discrepancy in MSE between the two algorithms diminishes. By the time the sample size reaches 500, the difference dwindles to a mere 2%. This observation suggests that while FMEE may lag slightly behind MEE, particularly in larger-scale datasets, such disparity remains acceptable in practical applications, given FMEE's significantly reduced computational time compared to MEE.

Table 2. The mean squared error of MEE and FMEE for Gaussian noise.

Sample Size	Average MSE		Optimal MSE	
	MEE	FMEE	MEE	FMEE
100	0.9108	1.4027	0.5697	0.6542
200	0.9311	1.0528	0.6295	0.7258
300	0.9573	1.0190	0.7600	0.7462
400	0.9696	0.9769	0.8024	0.7588
500	0.9604	0.9938	0.8024	0.8302

Table 3 analyses the iteration counts required for convergence between MEE and FMEE in the context of Gaussian noise. Notably, FMEE achieves convergence with significantly fewer iterations compared to MEE. Furthermore, it is observed that the iteration count of MEE gradually decreases as the sample size increases. Conversely, the iteration count of FMEE remains relatively constant, irrespective of variations in sample size.

Table 3. The number of iterations of MEE and FMEE for Gaussian noise.

Sample Size	Average Iteration Number		Optimal Iteration Number	
	MEE	FMEE	MEE	FMEE
100	797.01	38.21	592	6
200	697.09	38.29	592	7
300	656.61	36.88	542	7
400	650.03	40.67	571	9
500	637.03	36.17	586	7

Table 4 provides a comparative assessment of the time consumption between MEE and FMEE in the context of non-Gaussian noise. Across all five experimental groups, FMEE consistently demonstrates a remarkable reduction in time consumption, consistently amounting to less than 1% of MEE's time expenditure. Moreover, in accordance with theoretical predictions, the computational cost of MEE exhibits quadratic growth relative to the sample size, whereas FMEE's time overhead scales linearly with the sample size.

Table 4. The running time of MEE and FMEE for non-Gaussian noise.

Sample Size	Average Running Time		Optimal Running Time	
	MEE	FMEE	MEE	FMEE
100	107.96	0.0413	85.24	0.0156
200	348.66	0.0635	270.26	0.0312
300	800.50	0.0819	727.45	0.0468
400	1431.34	0.1027	1274.87	0.0624
500	2179.33	0.1178	1952.27	0.0624

Table 5 conducts a comparative evaluation of the mean squared error (MSE) between MEE and FMEE in the context of non-Gaussian noise. Generally, the disparity between the MSE values of FMEE and MEE is relatively minor. Notably, FMEE outperforms MEE when the sample size is 100 or 400, while MEE exhibits superior performance for sample sizes of 200, 300, and 500. Additionally, FMEE attains the optimal MSE across all sample sizes, except for $n = 100$. Furthermore, it is observed that the MSE values for non-Gaussian noise, as depicted in Table 5, span a broader range compared to those for Gaussian noise, indicating greater variability in results for both MEE and FMEE.

Table 5. The mean squared error of MEE and FMEE for non-Gaussian noise.

Sample Size	Average MSE		Optimal MSE	
	MEE	FMEE	MEE	FMEE
100	1.0844	0.9604	0.1584	0.1630
200	0.8485	1.1554	0.2654	0.2607
300	0.9110	1.2314	0.3140	0.2659
400	1.1037	1.0550	0.3672	0.2891
500	0.9536	1.0491	0.4995	0.4668

Table 6 compares the number of iterations required for convergence between MEE and FMEE in the context of non-Gaussian noise. FMEE consistently demonstrates faster convergence compared to MEE. Furthermore, it is observed that the iteration count of MEE decreases as the sample size increases. In contrast, the iteration count of FMEE remains relatively stable despite variations in sample size.

Table 6. The number of iterations of MEE and FMEE for non-Gaussian noise.

Sample Size	Average Iteration Number		Optimal Iteration Number	
	MEE	FMEE	MEE	FMEE
100	782.88	34.33	618	8
200	689.67	35.72	563	9
300	644.95	36.72	584	10
400	648.70	37.55	580	12
500	633.25	35.35	571	12

Based on the findings presented in Tables 1–6, it is evident that FMEE consistently yields comparable outcomes to MEE, albeit with significantly reduced time consumption and iteration counts. Notably, FMEE stands out for its remarkable efficiency, being approximately 1000 times faster than MEE and achieving convergence within approximately 0.1 s for 500 instances. Given its linear time complexity of $O(n)$, FMEE holds considerable promise for deployment in real-time scenarios characterized by non-Gaussian noise.

Subsequently, we embark on practical experimentation utilizing transformer characteristic data to forecast distribution network failures. The transformers are categorized into two groups based on their geographical location within distinct levels of urban activity within a specific city in northwest China. Initially, our attention is directed towards a subset of 1265 transformers situated within a smaller, less densely populated area of the aforementioned city. We undertake a random partitioning of this dataset into two segments: a training set comprising 80% of the data, totaling 1012 instances, and a verification set encompassing the remaining 20%, comprising 253 instances. To be more specific, the transformer dataset comprises various characteristic variables, involving the standardized heavy overload duration, maximum active load ratio, average active load ratio, mean three-phase unbalance, standardized heavy three-phase unbalance duration and so forth.

We conduct a comparative analysis, pitting our proposed FMEE against three established baseline methodologies: logistic regression, neural network, and support vector machine. The dataset comprises 1265 transformer characteristic variables associated with substantial load levels, undergoing 30 rounds of random partitioning. Subsequently, each randomly divided dataset is subjected to four distinct algorithms. The predictive efficacy of these algorithms on the verification set is assessed based on F-measure and error rate metrics.

Figure 1 depicts the F-measure evaluations stemming from 30 prediction instances conducted by four distinct algorithms. Notably, each evaluation involves consistent partitioning of the dataset across the four algorithms. A discernible trend emerges wherein all algorithms yield F-measure values surpassing 0.8. Notably, our proposed FMEE ex-

hibits the most promising performance, boasting an average F-measure of 0.904, thereby outshining the other three baseline methodologies by a significant margin.

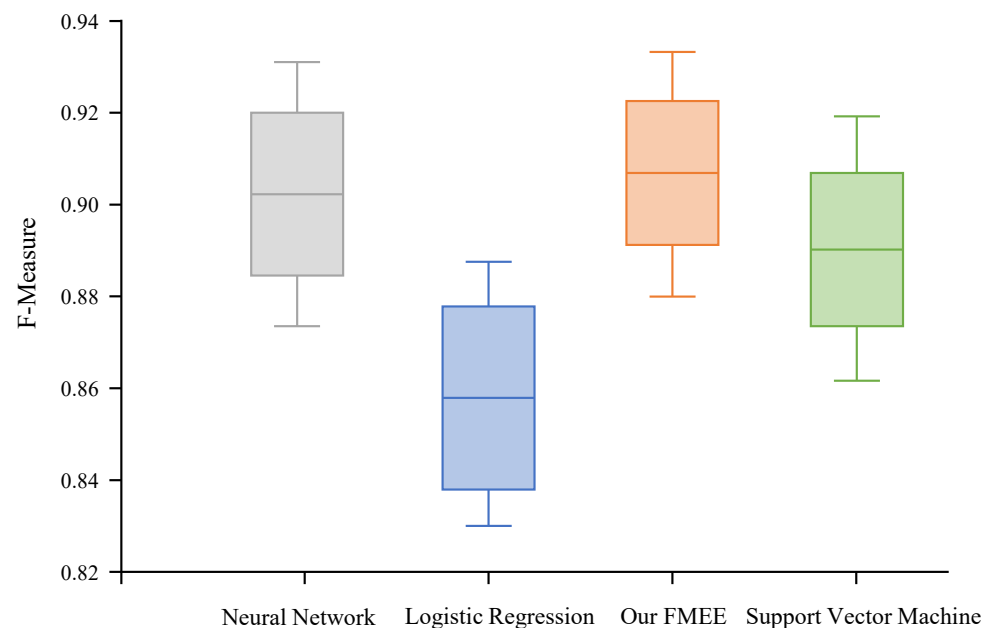


Figure 1. The boxplot of F-measure among four algorithms for failure outage prediction results.

To facilitate a comprehensive comparison among the four algorithms, Figure 2 depicts the error rate and F-measure of fault outage prediction results in the form of line and box charts. Notably, FMEE emerges as a frontrunner, showcasing a substantial advantage over the sophisticated support vector machine in terms of error rate. Specifically, our FMEE achieves an average error rate of 9.1%, surpassing the neural network, logistic regression, and support vector machine by margins of 0.3%, 4.2%, and 4.3%, respectively. These findings underscore the clear superiority of our proposed FMEE methodology.

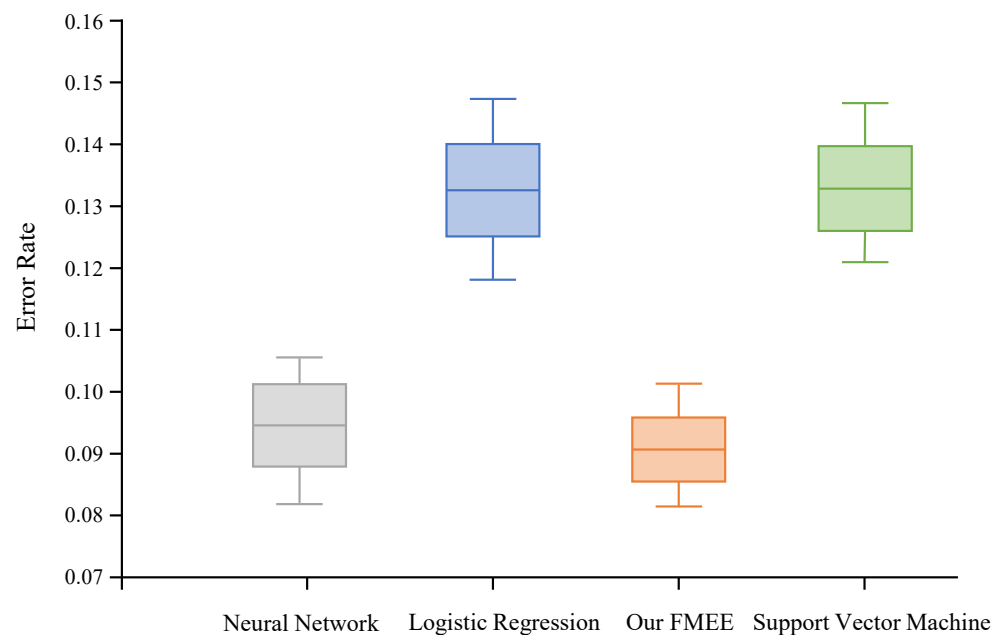


Figure 2. The boxplot of error rate among four algorithms for failure outage prediction results.

Furthermore, we extend our experimentation to a more densely populated urban region within the northwest Chinese city. Figure 3 illustrates the F-measure evaluations derived from 30 independent tests conducted across the four comparative algorithms. On a comprehensive scale, our proposed FMEE attains an F-measure of 0.891, surpassing baseline methodologies such as neural networks, logistic regression, and support vector machines by margins of 0.013, 0.023, and 0.028, respectively.

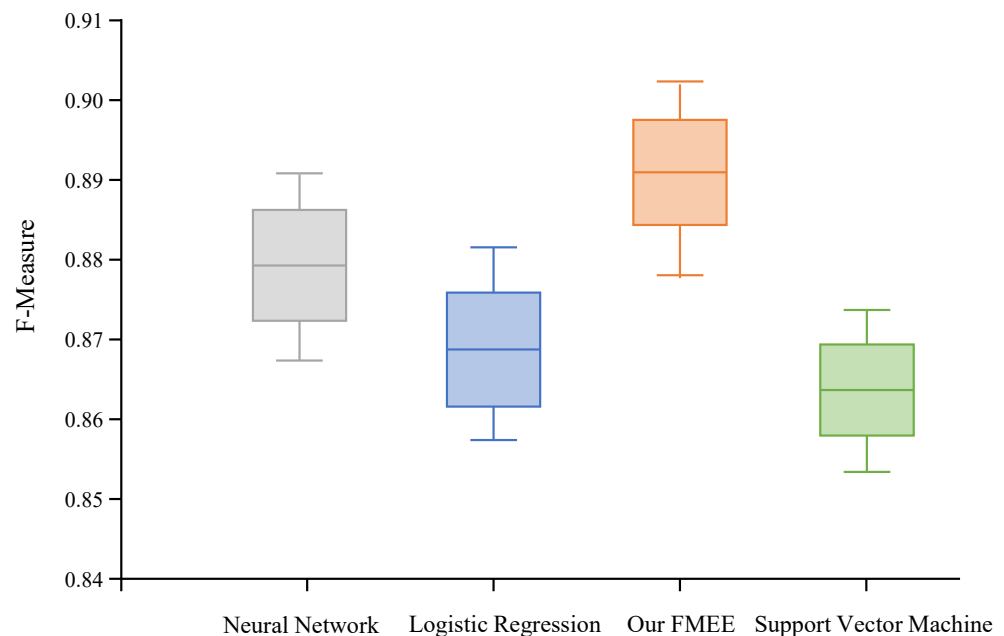


Figure 3. The boxplot of F-measure of four algorithms for failure outage prediction results.

Finally, we report the error rate of F-measure concerning fault outage prediction results derived from four comparative algorithms, represented through line and box charts. As shown in Figure 4, our proposed FMEE achieves an average error rate of 11.2%, showcasing a notable superiority over three baseline methodologies. These findings robustly underscore the efficacy of our algorithm in addressing regression problems.

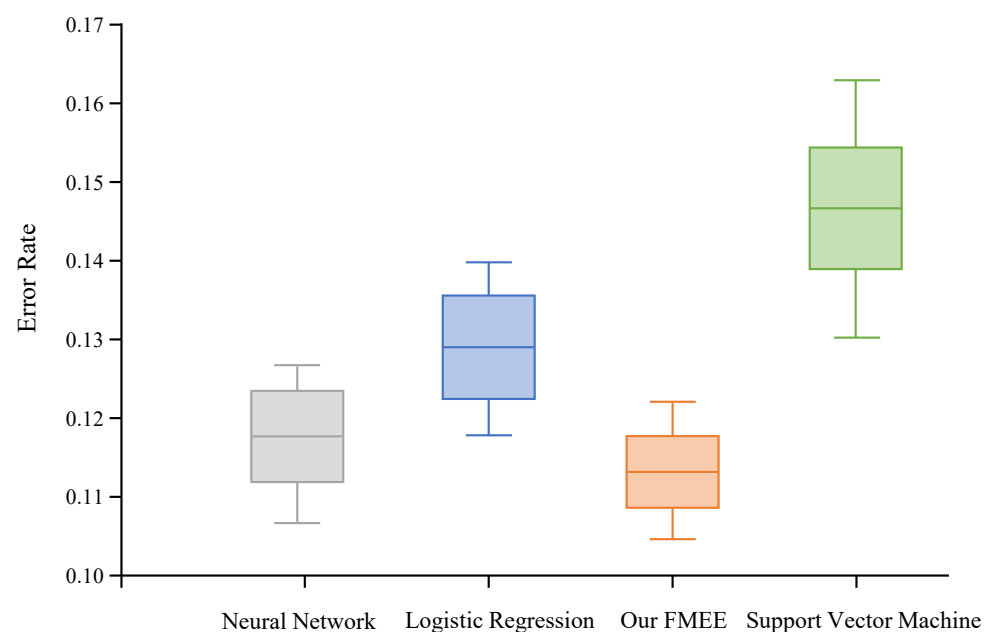


Figure 4. The boxplot of error rate of four algorithms for failure outage prediction results.

5. Conclusions

This study introduces a rapid minimum error entropy algorithm tailored for linear regression. Several notable conclusions emerge from our investigation. Firstly, the proposed FMEE exhibits significantly faster convergence, requiring fewer iterations compared to MEE. Secondly, while FMEE does not surpass traditional MEE in performance for Gaussian noise and yields comparable results for non-Gaussian noise as measured by mean squared error, its computational efficiency is strikingly superior, consuming less than 1% of the time required by MEE. Thirdly, despite being derived under the assumption of noise proximity to Gaussian distribution, FMEE demonstrates efficacy in handling non-Gaussian, heavy-tailed noise distributions. Fourthly, owing to its exceptional time efficiency, FMEE holds promise for real-time applications in scenarios featuring non-Gaussian noise. Moving forward, further analysis of FMEE's convergence conditions and exploration of noise approximation methods for arbitrary distributions represent avenues for future research.

Author Contributions: Conceptualization, Q.L. and X.L.; methodology, Q.L.; software, W.C.; validation, Q.L., X.L. and W.C.; formal analysis, Y.W.; investigation, X.L.; resources, Q.L.; data curation, Y.W.; writing—original draft preparation, Q.G.; writing—review and editing, H.C.; visualization, Q.G.; supervision, H.C.; project administration, W.C.; funding acquisition, Q.L. and H.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the State Grid Information and Telecommunication Group scientific and technological innovation projects “Research on Power Digital Space Technology System and Key Technologies” (Grant no: SGIT0000XMJS2310456).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available upon request from the corresponding author. The data are not publicly available due to the special significance of the electrical power system used in the experimental process, which involve privacy in some research fields, it is not convenient to fully disclose their data.

Conflicts of Interest: Authors (Qiang Li, Xiao Liao, Wei Cui, and Ying Wang) were employed by the State Grid Information & Telecommunication Group Co., Ltd. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Baggenstoss, P.M. Maximum Entropy PDF Design Using Feature Density Constraints: Applications in Signal Processing. *IEEE Trans. Signal Process.* **2015**, *11*, 2815–2825. [\[CrossRef\]](#)
2. Li, Y.; Bai, X.; Yuan, D.; Yu, C.; San, X.; Guo, Y.; Zhang, L.; Ye, J. Cu-based high-entropy two-dimensional oxide as stable and active photothermal catalyst. *Nat. Commun.* **2023**, *14*, 3171. [\[CrossRef\]](#) [\[PubMed\]](#)
3. Bisikalo, O.; Kharchenko, V.; Kovtun, V.; Krak, I.; Pavlov, S. Parameterization of the stochastic model for evaluating variable small data in the Shannon entropy basis. *Entropy* **2023**, *25*, 184. [\[CrossRef\]](#) [\[PubMed\]](#)
4. Chen, Q.; Zhang, F.; Su, L.; Lin, B.; Chen, S.; Zhang, Y. State Parameter Fusion Estimation for Intelligent Vehicles Based on IMM-MCCKF. *Appl. Sci.* **2024**, *14*, 4495. [\[CrossRef\]](#)
5. Carli, F.P.; Chen, T.; Ljung, L. Maximum entropy kernels for system identification. *IEEE Trans. Autom. Control* **2016**, *62*, 1471–1477. [\[CrossRef\]](#)
6. Fan, X.; Chen, L.; Huang, D.; Tian, Y.; Zhang, X.; Jiao, M.; Zhou, Z. From Single Metals to High-Entropy Alloys: How Machine Learning Accelerates the Development of Metal Electrocatalysts. *Adv. Funct. Mater.* **2024**, *1*, 2401887. [\[CrossRef\]](#)
7. Han, T.T.; Ge, S.S.; Lee, T.H. Persistent Dwell-Time Switched Nonlinear Systems: Variation Paradigm and Gauge Design. *IEEE Trans. Autom. Control* **2010**, *55*, 321–337.
8. Chesi, G. Stabilization and Entropy Reduction via SDP-Based Design of Fixed-Order Output Feedback Controllers and Tuning Parameters. *IEEE Trans. Autom. Control* **2017**, *62*, 1094–1108. [\[CrossRef\]](#)
9. Wei, P.; Li, H.X. Spatiotemporal entropy for abnormality detection and localization of Li-ion battery packs. *IEEE Trans. Ind. Electron.* **2023**, *70*, 12851–12859. [\[CrossRef\]](#)
10. Wang, Z.; Li, G.; Yao, L.; Cai, Y.; Lin, T.; Zhang, J.; Dong, H. Intelligent fault detection scheme for constant-speed wind turbines based on improved multiscale fuzzy entropy and adaptive chaotic Aquila optimization-based support vector machine. *ISA Trans.* **2023**, *138*, 582–602. [\[CrossRef\]](#) [\[PubMed\]](#)

11. Daneshpazhouh, A.; Sami, A. Entropy-based outlier detection using semi-supervised approach with few positive examples. *Pattern Recogn. Lett.* **2014**, *49*, 77–84. [\[CrossRef\]](#)
12. Qu, B.; Wang, Z.; Shen, B.; Dong, H. Decentralized dynamic state estimation for multi-machine power systems with non-Gaussian noises: Outlier detection and localization. *Automatica* **2023**, *153*, 111010. [\[CrossRef\]](#)
13. Feng, Z.; Wang, G.; Peng, B.; He, J.; Zhang, K. Novel robust minimum error entropy wasserstein distribution kalman filter under model uncertainty and non-gaussian noise. *Signal Process.* **2023**, *203*, 108806. [\[CrossRef\]](#)
14. Sun, M.; Davies, M.E.; Proudler, I.K.; Hopgood, J.R. Adaptive kernel Kalman filter. *IEEE Trans. Signal Process.* **2023**, *71*, 713–726. [\[CrossRef\]](#)
15. Wu, L.; Lin, H.; Hu, B.; Tan, C.; Gao, Z.; Liu, Z.; Li, S.Z. Beyond homophily and homogeneity assumption: Relation-based frequency adaptive graph neural networks. *IEEE Trans. Neural Netw. Learn. Syst.* **2023**, *35*, 8497–8509. [\[CrossRef\]](#) [\[PubMed\]](#)
16. Guo, L.; Wang, H. Minimum entropy filtering for multivariate stochastic systems with non-Gaussian noises. *IEEE Trans. Autom. Control* **2006**, *51*, 695–700.
17. Wang, Y.; Tang, Y.Y.; Li, L. Minimum error entropy based sparse representation for robust subspace clustering. *IEEE Trans. Signal Process.* **2015**, *63*, 4010–4021. [\[CrossRef\]](#)
18. Wang, M.; Deng, W. Deep face recognition: A survey. *Neurocomputing* **2021**, *429*, 215–244. [\[CrossRef\]](#)
19. Lin, M.; Cheng, C.; Peng, Z.; Dong, X.; Qu, Y.; Meng, G. Nonlinear dynamical system identification using the sparse regression and separable least squares methods. *J. Sound Vib.* **2021**, *505*, 116141. [\[CrossRef\]](#)
20. Wu, Z.; Peng, S.; Ma, W.; Chen, B.; Principe, J.C. Minimum error entropy algorithms with sparsity penalty constraints. *Entropy* **2015**, *17*, 3419–3437. [\[CrossRef\]](#)
21. Huang, S.; Tran, T.D. Sparse signal recovery via generalized entropy functions minimization. *IEEE Trans. Signal Process.* **2018**, *67*, 1322–1337. [\[CrossRef\]](#)
22. Yue, H.; Wang, H. Minimum entropy control of closed-loop tracking errors for dynamic stochastic systems. *IEEE Trans. Autom. Control* **2003**, *48*, 118–122.
23. Wang, H. Minimum entropy control of non-Gaussian dynamic stochastic systems. *IEEE Trans. Autom. Control* **2002**, *47*, 398–403. [\[CrossRef\]](#)
24. Mitra, R.; Bhatia, V. Minimum error entropy criterion based channel estimation for massive-MIMO in VLC. *IEEE Trans. Veh. Technol.* **2018**, *68*, 1014–1018. [\[CrossRef\]](#)
25. Erdogmus, D.; Principe, J.C. Convergence properties and data efficiency of the minimum error entropy criterion in adaline training. *IEEE Trans. Signal Process.* **2003**, *51*, 1966–1978.
26. Chen, B.; Yuan, Z.; Zheng, N.; Principe, J.C. Kernel minimum error entropy algorithm. *Neurocomputing* **2013**, *121*, 160–169. [\[CrossRef\]](#)
27. Hu, T.; Wu, Q.; Zhou, D.X. Convergence of gradient descent for minimum error entropy principle in linear regression. *IEEE Trans. Signal Process.* **2016**, *64*, 6571–6579. [\[CrossRef\]](#)
28. Zhang, Y.; Chen, B.; Liu, X.; Yuan, Z.; Principe, J.C. Convergence of a fixed-point minimum error entropy algorithm. *Entropy* **2015**, *17*, 5549–5560. [\[CrossRef\]](#)
29. Chen, B.; Zhu, Y.; Hu, J.; Zhang, M. On optimal estimations with minimum error entropy criterion. *J. Frankl. Inst.* **2010**, *347*, 545–558.
30. Heravi, A.R.; Hodtani, G.A. A new information theoretic relation between minimum error entropy and maximum correntropy. *IEEE Signal Process. Lett.* **2018**, *25*, 921–925.
31. Yang, S.; Tan, J.; Chen, B. Robust spike-based continual meta-learning improved by restricted minimum error entropy criterion. *Entropy* **2022**, *24*, 455. [\[CrossRef\]](#) [\[PubMed\]](#)
32. Dang, L.; Chen, B.; Wang, S.; Ma, W.; Ren, P. Robust power system state estimation with minimum error entropy unscented Kalman filter. *IEEE Trans. Instrum. Meas.* **2020**, *69*, 8797–8808. [\[CrossRef\]](#)
33. Wang, B.; Hu, T. Semi-Supervised Minimum Error Entropy Principle with Distributed Method. *Entropy* **2018**, *20*, 968. [\[CrossRef\]](#) [\[PubMed\]](#)
34. Chen, B.; Xing, L.; Xu, B.; Zhao, H.; Principe, J.C. Insights into the robustness of minimum error entropy estimation. *IEEE Trans. Neural Netw. Learn. Syst.* **2016**, *29*, 731–737. [\[CrossRef\]](#) [\[PubMed\]](#)
35. Shen, P.; Li, C. Minimum total error entropy method for parameter estimation. *IEEE Trans. Signal Process.* **2015**, *63*, 4079–4090. [\[CrossRef\]](#)
36. Kothari, P.; Kreiss, S.; Alahi, A. Human trajectory forecasting in crowds: A deep learning perspective. *IEEE Trans. Intell. Transp. Syst.* **2021**, *23*, 7386–7400. [\[CrossRef\]](#)
37. Kim, N.; Kwon, K. Normalized Minimum Error Entropy Algorithm with Recursive Power Estimation. *Entropy* **2016**, *18*, 239. [\[CrossRef\]](#)
38. Chen, B.; Ma, R.; Yu, S.; Du, S.; Qin, J. Granger causality analysis based on quantized minimum error entropy criterion. *IEEE Signal Process. Lett.* **2019**, *26*, 347–351. [\[CrossRef\]](#)
39. Niknam, M.; Santos, L.F.; Cory, D.G. Experimental detection of the correlation Rényi entropy in the central spin model. *Phys. Rev. Lett.* **2021**, *127*, 080401. [\[CrossRef\]](#) [\[PubMed\]](#)
40. Kim, T.; San Kim, D.; Jang, L.C.; Lee, H.; Kim, H. Representations of degenerate Hermite polynomials. *Adv. Appl. Math. Mech.* **2022**, *139*, 102359. [\[CrossRef\]](#)

41. Lin, W.; Zhang, J.E. The valid regions of Gram–Charlier densities with high-order cumulants. *J. Comput. Appl. Math.* **2022**, *407*, 113945. [[CrossRef](#)]
42. Matsushita, R.; Brandão, H.; Nobre, I.; Da Silva, S. Differential entropy estimation with a Paretian kernel: Tail heaviness and smoothing. *Phys. A Stat. Mech. Appl.* **2024**, *1*, 129850. [[CrossRef](#)]
43. Canales, C.; Galarce, C.; Rubio, F.; Pineda, F.; Anguita, J.; Barros, R.; Parragué, M.; Daille, L.K.; Aguirre, J.; Armijo, F.; et al. Testing the test: A comparative study of marine microbial corrosion under laboratory and field conditions. *ACS Omega* **2021**, *6*, 13496–13507. [[CrossRef](#)] [[PubMed](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.

Article

Optimization of Gene Selection for Cancer Classification in High-Dimensional Data Using an Improved African Vultures Algorithm

Mona G. Gafar ^{1,2,*} , Amr A. Abohany ^{3,†} , Ahmed E. Elkhoul ^{4,†} and Amr A. Abd El-Mageed ^{5,†} ¹ Department of Computer Engineering and Information, College of Engineering in Wadi Alddawasir, Prince Sattam bin Abdulaziz University, Kharj 16278, Saudi Arabia² Machine Learning and Information Retrieval Department, Artificial Intelligence, Kafrelsheikh University, Kafrelsheikh 33511, Egypt³ Faculty of Computers and Information, Kafrelsheikh University, Kafrelsheikh 33511, Egypt; amrabohany8@gmail.com⁴ Department of Biomedical Engineering, Faculty of Electrical Engineering, Menofia University, Menofia 32951, Egypt; eng.elkholy.ahmed@gmail.com⁵ Department of Information Systems, Sohag University, Sohag 82511, Egypt; amr.atef@commerce.sohag.edu.eg

* Correspondence: m.gafar@psau.edu.sa or dr.mona.gamal85@gmail.com

† These authors contributed equally to this work.

Abstract: This study presents a novel method, termed RBAVO-DE (Relief Binary African Vultures Optimization based on Differential Evolution), aimed at addressing the Gene Selection (GS) challenge in high-dimensional RNA-Seq data, specifically the rnaseqv2 illuminaHiSeq rnaseqv2 un edu Level 3 RSEM genes normalized dataset, which contains over 20,000 genes. RNA Sequencing (RNA-Seq) is a transformative approach that enables the comprehensive quantification and characterization of gene expressions, surpassing the capabilities of micro-array technologies by offering a more detailed view of RNA-Seq gene expression data. Quantitative gene expression analysis can be pivotal in identifying genes that differentiate normal from malignant tissues. However, managing these high-dimensional dense matrix data presents significant challenges. The RBAVO-DE algorithm is designed to meticulously select the most informative genes from a dataset comprising more than 20,000 genes and assess their relevance across twenty-two cancer datasets. To determine the effectiveness of the selected genes, this study employs the Support Vector Machine (SVM) and k-Nearest Neighbor (*k*-NN) classifiers. Compared to binary versions of widely recognized meta-heuristic algorithms, RBAVO-DE demonstrates superior performance. According to Wilcoxon's rank-sum test, with a 5% significance level, RBAVO-DE achieves up to 100% classification accuracy and reduces the feature size by up to 98% in most of the twenty-two cancer datasets examined. This advancement underscores the potential of RBAVO-DE to enhance the precision of gene selection for cancer research, thereby facilitating more accurate and efficient identification of key genetic markers.

Keywords: RNA sequencing; gene selection; high dimensionality; meta-heuristic algorithms; African vultures algorithm; differential evolution; relief technique



Citation: Gafar, M.G.; Abohany, A.A.; Elkhoul, A.E.; Abd El-Mageed, A.A. Optimization of Gene Selection for Cancer Classification in High-Dimensional Data Using an Improved African Vultures Algorithm. *Algorithms* **2024**, *17*, 342. <https://doi.org/10.3390/a17080342>

Academic Editors: Muhammad Adnan Khan and Frank Werner

Received: 27 April 2024

Revised: 16 July 2024

Accepted: 1 August 2024

Published: 6 August 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Deoxyribonucleic Acid (DNA) makes up our genetic code, containing the recipe for our existence. Although every cell has the same DNA, each tissue structure is unique and serves a distinct purpose. The RNA transcription mechanism determines which genes in a cell are active, enabling RNA to be transformed into proteins responsible for the cell's structure and functionality [1]. We analyze RNA-Seq gene expression data (GED) to determine genetic changes and evaluate disease biomarkers. Differential expression analysis identifies quantitative distinctions in gene expression, allowing us to categorize genes whose expression changes under various conditions. This method helps us understand

illnesses and find ways to manage them. Gene expression profiling technologies have significantly developed over the years [2].

Two popular techniques are micro-array, which uses a hybridization-based approach, and RNA-Seq, which is based on next-generation sequencing. These technologies have demonstrated their value in advancing our understanding of gene expression [3]. Notable studies, such as those by Chen et al. [4] and Nunez et al. [5], have highlighted the potential of these techniques, while Wang et al. [6] have provided valuable insights into the RNA-Seq approach.

The two methods under consideration serve the purpose of quantifying gene expression for classification and statistical investigation. This paper has chosen the quantification data obtained through the next-generation sequencing (NGS)-based RNA-Seq approach because it offers more accuracy in detecting RNA quantification levels compared to micro-array data [7]. RNA-Seq has overcome several limitations of micro-array analysis, such as dependence on prior sequencing knowledge, which restricted the detection range [8,9]. Unlike micro-arrays, RNA-Seq does not require any previous knowledge and significantly expands the dynamic detection range. This enhancement improves the accuracy of results and enables the identification of a comprehensive set of genes, providing a precise understanding of disease biomarkers. As a result, RNA-Seq has emerged as a potent tool for researchers, offering valuable insights into the underlying molecular mechanisms of various diseases [10].

The “dimensionality curse” [11] has become a common challenge in contemporary times owing to the abundance of available data. This has caused a surge in the development of feature selection (FS) techniques and algorithms. FS algorithms can be classified into four distinct methods: filter, wrapper, embedded, and hybrid [12,13]. These approaches aim to identify the most informative features that can distinguish between different classes. In our case, we are concerned with identifying the genes linked to tumors.

The filter approach is a widely used technique in FS that involves evaluating the relevance of individual genes based on statistical scores. This method is known for its high accuracy in selecting the most satisfactory group of genes. Nevertheless, the filter approach has some limitations, as functioning on each gene individually ignores the interrelationships between genes, which can result in a local optima issue [14]. It is worth noting that the filter approach can be categorized into two sub-types—univariate and multivariate—with the latter considering the correlations between genes. Some examples of the filter approach include Relief [15], Fisher score [16], *t*-test [17], and information gain [18].

The wrapper method investigates all feasible gene subsets to test and create a subset of genes to determine their implications. A specific classifier is employed to determine the outcome of each subset, and the categorization technique is used multiple times for each assessment. Compared to the filter approach, the wrapper approach delivers superior performance by using a categorization technique that directs the learning procedure. However, this method requires substantial time and computational resources, especially when dealing with large-scale data [19].

Meta-heuristic methods (MHMs) are high-level heuristics used in mathematical optimization and computer science to find satisfactory solutions to optimization problems when information is imperfect or resources are limited [20]. MHMs sample a subset of solutions, making them useful for various optimization issues due to their few assumptions about the problem [21]. However, they do not guarantee globally optimal solutions. Many MHMs employ stochastic optimization, meaning the solution is based on random variables generated during the process [22].

Meta-heuristic methods (MHMs) are more practical than traditional iterative methods and optimization algorithms because they can explore a broader range of possible solutions. Consequently, MHMs have become a preferred strategy for solving optimization problems [23]. Numerous research papers have demonstrated that among the different wrapper methods, MHMs are well suited to address the feature selection (FS) issue. Stochastic methods, including MHMs, can produce optimal or near-optimal results quickly. They

offer benefits such as flexibility, self-management without detailed mathematical properties, and the ability to assess numerous outcomes simultaneously. Various MHMs have recently been developed to solve the FS problem, providing reliable near-optimal solutions at significantly reduced computational costs [24].

Embedded FS methods utilize a learning technique to select appropriate genes that interact with the classification process. This method combines the FS technique as part of the learning model. The learning algorithm is trained with an initial attribute subset to estimate a measure for evaluating the rank values of attributes [25]. The ultimate goal is to decrease the computation time for reclassifying different subsets by incorporating the FS stage into the training process. Some techniques in this approach perform feature weighting based on regularization models with objective functions that minimize fitting errors while enforcing feature coefficients to be small or precisely zero. Examples of embedded methods are the First-Order Inductive Learner (FOIL) rule-based feature subset selection algorithm and SVM based on Recursive Feature Elimination (SVM-RFE) [26].

The hybrid method is a well-crafted technique that amalgamates the filter and wrapper methods to leverage the strengths of each. This approach begins with a filter that reduces the feature space dimensionality, generating multiple subsets with intermediate complexity. Subsequently, a wrapper is employed as a learning technique to choose the most suitable candidate subset [27]. The hybrid approach integrates the accuracy of wrappers and the efficiency of filters, resulting in an optimal methodology [28,29].

1.1. Motivation and Contributions

An effective global meta-heuristic optimization algorithm is presented in this paper, which is called the African Vultures Optimization (AVO) algorithm [30], which imitates the living and eating habits of African vultures. The proposed algorithm consists of four basic phases: population division into three groups—the best solution, the second-best solution, and the remaining solutions; famine-level measurement to formulate a mathematical model for the vultures' exploitation, exploration, and transfer; exploration of vultures employing two strategies to allow vultures to cover large distances over extended periods to find food at random sites; and exploitation of vultures involving two sub-phases. Two distinct strategies are used in the first sub-phase: siege fight for strong vultures and rotational flight, which forms a spiral activity between the outstanding vulture and the rest. Two strategies are utilized in the second sub-phase: assembling vultures around the food source and a hostile siege fight, where vultures become more aggressive and attempt to steal food left behind by healthy vultures.

This study proposes an enhanced binary variant of the AVO technique, namely the RBAVO-DE technique, which is a productive approach that demonstrates accurate performance in handling the GS problem. At first, there is a high possibility that the recommended technique will steer clear of local optima and accomplish sufficient examination precision, quick convergence, and improved stability. Compared to recent MHMs, the proposed RBAVO-DE technique achieves enhanced efficacy by producing optimal or substantially optimal solutions for many of the analyzed situations. The Relief algorithm is utilized to identify only the related features to produce the final classification dataset. RBAVO-DE combines the Relief algorithm with the DE approach to increase exploration capability and obtain the best results within the solution space via repetitions. It also utilizes a transfer function to transform real position values into binary values. The RBAVO-DE approach makes sense in GS because it is simple to comprehend and implement, can deal with various optimization problems, produces valuable results in an acceptable amount of time, requires less computing power, and uses a small number of control parameters. This paper's primary contributions are as follows:

- Level 3 data based on next-generation sequencing (RNA-Seq) have been pre-processed.
- The ability of the proposed AVO meta-heuristic algorithm to address the GS problem has not been investigated, as RNA-Seq GED has never been used with the AVO algorithm.

- To construct a binary version known as the RBAVO-DE algorithm, the AVO algorithm is altered and recreated.
- The proposed RBAVO-DE algorithm combines the binary variant of the AVO with a Relief approach and DE to improve the exploration ability of the search space and enhance the achieved optimal results.
- The proposed RBAVO-DE algorithm is applied to RNA-Seq GED for the first time.
- Several performance indicators, including average fitness, classification accuracy, number of selected genes, precision, recall, and F1-score, are used to assess the outcomes.
- A comparison is made between the impact of the presented RBAVO-DE algorithm, employing the two recommended ML classifiers (SVM and k -NN), and other algorithms in the literature.
- Twenty-two distinct cancer datasets are used to assess the proposed RBAVO-DE method, and the results are shown.
- The chosen genes are investigated using biomarkers associated with cancer.

1.2. Structure

This paper is structured into five main sections. Section 2 reviews previous research on FS using RNA-Seq GED. This is followed by Section 3, which offers an in-depth discussion of the RBAVO-DE algorithm, an enhanced version of AVO, including its parameters for addressing GS. Section 4 showcases the experimental findings compared to several recent MHMs. Lastly, Section 6 concludes this study and proposes avenues for future investigation.

2. Related Works

In this section, we discuss the literature that focuses on the methods researchers use to classify RNA-Seq GED, which typically has high dimensionality. To achieve optimal performance of classification methods, it is crucial to disregard irrelevant and unrelated genes; hence, selecting appropriate genes is a vital stage before utilizing ML and deep learning (DL) techniques [31] or any other classification techniques. In this regard, we have explored some relevant papers in this domain to accomplish the objective of RNA-Seq categorization for cancer identification.

Yaqoob et al. [32] introduced a cutting-edge technique for GS known as the Sine-Cosine-Cuckoo Search Algorithm (SCCSA), a hybrid method tailored to function alongside established ML models such as SVM. This innovative GS algorithm was evaluated using a breast cancer benchmark dataset, where its performance was meticulously analyzed and compared with other GS methodologies. To refine the selection of features, the minimum Redundancy Maximum Relevance method was initially applied as a preliminary filtering step. Subsequently, the hybrid SCCSA approach was deployed to further improve and fine-tune the GS process. The final stage involved using the SVM classifier to classify the dataset based on the selected genes. Considering the critical importance of GS in decoding complex biological datasets, SCCSA emerges as a crucial asset for the classification of cancer-related datasets.

Joshi et al. [33] introduced an innovative optimization strategy named PSO-CS integrated with DL for brain tumor classification. This method enhances the efficiency of the Particle Swarm Optimization (PSO) technique by incorporating the Cuckoo Search (CS) algorithm, optimizing the classification process. Following this optimization, PSO-CS utilizes DL to classify GED related to brain tumors, identifying various classes associated with specific tumors alongside the PSO-CS optimization method. By integrating the PSO-CS technique with DL, it significantly outperformed other DL and ML models regarding classification accuracy, as evidenced by various performance measures.

Mahto et al. [34] unveiled a groundbreaking approach for cancer classification through an integrated method based on CS-SMO (CS and Spider Monkey Optimization) for GS. Initially, the fitness function of the Spider Monkey Optimization (SMO) algorithm is modified using the CS algorithm. This modification leverages the strengths of both MHMs

to identify a subset of genes capable of predicting cancer at an early stage. To further refine the accuracy of the CS-SMO algorithm, a pre-processing step, MRMR, is employed to reduce the complexity of cancer gene expression datasets. Subsequently, these gene subsets are processed using DL to classify different cancer types. The efficacy of the CS-SMO approach coupled with DL was evaluated using eight benchmark micro-array gene expression datasets for cancer, examining its performance across various measures. The CS-SMO method integrated with DL demonstrated superior classification accuracy across all examined large-scale gene expression datasets for cancer, outperforming existing DL and ML models.

Neggaz et al. [35] introduced an improved version of the manta ray foraging optimization, called MRFO-SC, which used trigonometric operators inspired by the sine-cosine (SC) algorithm to handle the GS issue. The k -NN model was used for gene-set selection. In addition, the statistical significance of the MRFO-SC was evaluated using Wilcoxon's rank-sum test at a 5% significance level. The results were evaluated and compared with some recent MHMs. The comparison and experimental results confirmed the effective performance of the proposed MRFO-SC on high- and low-dimensional benchmark datasets by obtaining the greatest classification accuracy on 85% of the GS benchmark datasets.

Lyu et al. [36] explored cancer biomarkers by focusing on genes' significance in their impact on classification. They followed a two-phase approach—data pre-processing and utilizing a convolutional neural network—to classify the type of tumor. In the second phase, they created heat maps for each category to identify genes related to pixels with the highest intensities in the heat maps. They then evaluated the selected genes' pathways. During pre-processing, they removed the levels of gene expression that had not been modified during the GS phase, using a variance threshold of 1.19, which decreased the number of genes from 19,531 to 10,381. The final classification accuracy obtained was 95.59, which is good but could still be improved using a more effective FS methodology to further decrease data dimensionality.

Khalifa et al. [37] built on the aforementioned paper [36]. The focus of their study was on five types of cancer data: Uterine Corpus Endometrial Carcinoma (UCEC), Lung Squamous Cell Carcinoma (LUSC), Lung Adenocarcinoma (LUAD), Kidney Renal Clear Cell Carcinoma (KIRC), and Breast Invasive Carcinoma (BRCA). The benchmark dataset used for the study comprised 2086 records and 972 attributes. Each record provided detailed sample information, while each attribute included the RNA-Seq values for a specific gene, represented as RPKM (Reads Per Kilobase per Million) [38]. The researchers employed a mixed approach using binary PSO with the decision trees (BPSO-DT) method to pre-process the data. Out of 971 attributes, 615 were selected as the best attributes of RNA-Seq. The proposed method achieved an overall testing classification accuracy of 96.90%, as demonstrated by the suggested outcomes and evaluation measures used.

Xiao et al. [39] assessed their methodology using three RNA-Seq datasets: Stomach Adenocarcinoma (STAD), BRCA, and LUAD. Their approach relied on DL techniques, wherein they employed five classifiers and subsequently utilized the DL technique to ensemble each output of the five classifiers. This led to an improvement in the classification accuracy of all the predictions, with the BRCA dataset achieving 98.4% accuracy, STAD achieving 98.78% accuracy, and LUAD achieving 99.20% accuracy.

Liu et al. [40] used micro-array data and a hybrid approach to address the problem. Unlike the previously mentioned papers, they studied each cancer class separately. To assess performance, they employed four gene benchmarks related to small round blue cell tumors, colon cancer, lung cancer, and leukemia. Their proposed method used Relief as the pre-processing technique to eliminate genes with a lower correlation with the specific cancer class, followed by PSO as the search technique. Finally, they employed the SVM model to evaluate the classification accuracy of the selected subset of genes and obtain the conclusive optimum gene subset for each cancer type.

Based on the existing research, it seems that most studies using RNA-Seq GED are still in their early stages, with researchers attempting to implement and test various ideas

in this promising area. While the literature contains a plethora of experiments employing multiple techniques, such as FS and DL recent methods, no single technique is perfect due to the high dimensionality of RNA-Seq GED. FS of RNA-Seq GED plays a crucial role in determining the relationship between a gene and its category. It is a critical pre-processing task for validating gene biomarkers of cancer and overcoming the dimensionality curse. Consequently, this study aims to introduce a new wrapper approach, the RBAVO-DE algorithm, and apply it to RNA-Seq data for the first time. It also compares the proposed algorithm's effectiveness with that of other FS techniques.

3. Proposed RBAVO-DE for GS

An improved variant of AVO known as RBAVO-DE is proposed in this paper to discover the smallest relevant gene subsets from the classification process and to disregard irrelevant genes. RBAVO-DE utilizes a Relief-based binary AVO algorithm combined with the DE technique. RBAVO-DE's primary feature is its ability to maximize accuracy while utilizing the fewest features possible. The proposed RBAVO-DE consists of two primary steps. First, there is a pre-processing step in which the Relief algorithm is used to determine which features are significant by assigning each feature a weight and then removing the features that are irrelevant and have the lowest weights. In the second step, the binary AVO algorithm and the DE technique are applied to identify the more pertinent and unique features. The AVO algorithm is prone to the local optimum trap when handling large-scale problems. The AVO algorithm incorporates the DE technique to prevent this.

The proposed RBAVO-DE algorithm for tackling the GS strategy requires applying the Relief algorithm, initializing, position boosting using the AVO algorithm, binary conversion, fitness appraisal, and integration with DE. The following subsections explain these steps.

3.1. Applying the Relief Algorithm for Feature Filtration

This step aims to pre-process the population using the Relief algorithm [41], which is considered a fast, easy, and efficient filtering technique for finding features related to one another. This algorithm's primary goal is to find characteristics that differentiate sample values and group similar samples close together. As a result, the method depends on the weighted ranking of features, where a feature with a higher weight indicates better classification performance.

After choosing a sample randomly, the Relief algorithm examines two different kinds of closest samples: near-hit samples, which are associated with samples from the same class, and near-miss samples, which are associated with samples from other classes. The near-hit and near-miss values can be used to determine the features' weights. To assess the importance in the classification process, the features' weights are arranged from most significant to least significant. Finally, the features with the most significant weights are selected. The following formula can be used to determine the weight W for feature A :

$$W_A = \sum_{j=1}^N \left(x_A^j - NM(x^j)_A \right)^2 - \left(x_A^j - NH(x^j)_A \right)^2. \quad (1)$$

where W_A denotes the weight of feature A , x_A^j denotes the value of feature A for data x^j , and N denotes the number of samples. The nearest data points to x^j in the same or different classes are $NH(x^j)$ and $NM(x^j)$.

The Relief algorithm narrows its focus to only the necessary features and minimizes the search area to help the AVO algorithm find better features more quickly.

3.2. Initializing the Population

The proposed BAVO algorithm starts by randomly generating a population of N positions. A vector of dimensions D equivalent to the number of features in the original dataset describes each position as a possible solution at its constrained lower and upper

bounds. This randomly initialized step, bounded within the $[-1, 1]$ range for each position vector's variable, uses the randomly produced position.

3.3. Boosting Positions via the AVO Algorithm

This paper presents the AVO algorithm [30], a meta-heuristic optimization algorithm inspired by vultures in Africa and their living and feeding behaviors. Based on fundamental ideas about vultures, the AVO algorithm is configured as follows: The African vulture population size is initially presumed by the AVO algorithm to consist of N vultures. The population of African vultures is then divided into three groups to reflect the primary natural function of vultures, based on the computed fitness function. The first group consists of the strongest vulture, which is the best solution; the second group contains a vulture weaker than the first, which is the second-best solution; and the final group encompasses the remaining weaker vultures, which are the worst solutions.

Based on the above, the proposed AVO algorithm is composed of four fundamental phases to simulate the behavior of different types of vultures. The following subsections clarify these phases.

3.3.1. Phase of Dividing the Population

The initial population is split into groups by assessing the solution's fitness function. The best solution is chosen as the best vulture in the first group, while the second group contains the second-best solution. The third group contains the remaining solutions. As the solutions constantly strive to approach the best and second-best solutions, the population needs to be re-evaluated for each iteration, as follows:

$$R^g = \begin{cases} BestVulture_1^g, & \text{if } pr^g = L_1, \\ SecondBestVulture_2^g, & \text{if } pr^g = L_2, \end{cases} \quad (2)$$

$$pr^g = \frac{v^g}{\sum_{g=1}^N v^g}. \quad (3)$$

The best vulture in the first group and the second-best vulture in the second group at the g^{th} iteration are denoted by the expressions $BestVulture_1^g$ and $SecondBestVulture_2^g$, respectively. The probability of selecting the best solution for each group at the g^{th} iteration is defined using the roulette wheel approach, as shown in Equation (3), pr^g . Two random parameters within the range $[0, 1]$ are L_1 and L_2 .

3.3.2. Phase of Measuring the Famine Level

This phase is utilized to formulate a mathematical model for the exploitation, exploration, and transfer among the vultures. When they are not famished, the vultures have a greater ability to search for food and can fly further. Furthermore, when famished, vultures cannot fly long distances in search of food and may become violent. The i^{th} vulture's famine level (F_i^g) during the g^{th} iteration can be written as follows, which is used to develop a mathematical model for the vultures' exploitation, exploration, and transfer:

$$F_i^g = (2 \times rand + 1) \times z \times \left(1 - \frac{g_i}{G_{max}}\right) + t, \quad (4)$$

$$t = h \times \left(\sin^w \left(\frac{\pi}{2} \times \frac{g_i}{G_{max}}\right) + \cos \left(\frac{\pi}{2} \times \frac{g_i}{G_{max}}\right) - 1\right). \quad (5)$$

The vultures are presumably full since the variable F_i^g shows the vultures' shift from exploration to exploitation. A random number between 0 and 1 is called a *rand*. z denotes a random value within the interval $[-1, 1]$. The current iteration number is denoted by g_i , while the maximum iteration number is denoted by G_{max} . Equation (5) calculates the t value to help solve complicated optimization problems more effectively and prevent reaching a local optimum. An arbitrary value within the interval $[-2, 2]$ is represented by h . The probability of carrying out the exploration process is regulated by the predefined

constant parameter w ; the exploration likelihood grows as its value increases. Exploration is less probable as its value declines.

Equation (4) states that as the number of iterations increases, F_i^g gradually lessens. As a result, the next step in the proposed AVO algorithm can be defined as follows:

$$\begin{cases} \text{Exploration phase (Search food in diverse areas),} & \text{if } |F_i^g| \geq 1, \\ \text{Exploitation phase (Search food in surrounding area),} & \text{if } |F_i^g| < 1. \end{cases} \quad (6)$$

3.3.3. Phase of Exploration

The vultures are characterized by their great ocular ability to locate appropriate food during this phase. There are two different strategies used in this phase to enable vultures to travel great distances for lengthy periods to search for food in random locations. These locations are selected using a random number $rand_{P_1}$ and a preset parameter P_1 , both of which have values within the range $[0, 1]$. In the exploration phase, the famine level $|F_i^g|$ is greater than or equal to 1. The exploration techniques are described as follows:

$$X_i^{g+1} = \begin{cases} R^g - D_i^g \times F_i^g, & \text{if } rand_{P_1} \leq P_1, \\ R^g - F_i^g + rand \times ((UB - LB) \times rand + LB), & \text{if } rand_{P_1} > P_1, \end{cases} \text{ if } |F_i^g| \geq 1, \quad (7)$$

$$D_i^g = |(2 \times rand) \times R^g - X_i^g|. \quad (8)$$

where the upcoming updated position at the next $(g + 1)^{th}$ iteration is denoted by X_i^{g+1} . The best vulture chosen for the current iteration g is denoted by R^g . This is determined using Equation (2). The vultures move randomly to protect the food from other vultures and to provide a high degree of randomness in their search behavior. $rand$ is a random value between zero and one. The variables' upper bound is denoted by UB ; their lower bound by LB ; and the current position at the g^{th} iteration is X_i^g .

3.3.4. Phase of Exploitation

In this phase, $|F_i^g|$ is less than 1. The proposed AVO algorithm's efficacy is evaluated in two sub-phases that comprise the exploitation phase. Each of these sub-phases uses two different strategies. For each internal sub-phase, the appropriate strategy is chosen using two predefined parameters: P_2 for the first sub-phase and P_3 for the second, with values ranging from 0 to 1. These two internal sub-phases are explained as follows:

- **First sub-phase of exploitation:** This sub-phase applies two different strategies when $|F_i^g|$ is less than 1 and greater than or equal to 0.5. The selection of one of these two strategies is made using a random value $rand_{P_2}$ within the range $[0, 1]$ and a specified parameter P_2 .

The initial strategy of this sub-phase is called **siege fight**, which involves sufficiently powerful and somewhat satiated vultures. More robust and healthier vultures try not to share food with others because they convene around a single food source. By swarming near the healthy vultures and engaging in small fights, the weaker vultures try to take food from them. Conversely, the second strategy is called **rotational flight**; it creates a spiral movement between a superior vulture and the others. The following illustrates the strategies for the first exploitation sub-phase:

$$X_i^{g+1} = \begin{cases} D_i^g \times (F_i^g + rand) - d_t^g, & \text{if } rand_{P_2} \leq P_2, \\ R^g - (S_1^g + S_2^g), & \text{if } rand_{P_2} > P_2, \end{cases} \text{ if } 1 > |F_i^g| \geq 0.5, \quad (9)$$

$$d_t^g = R^g - X_i^g, \quad (10)$$

$$S_1^g = R^g \times \left(\frac{rand \times X_i^g}{2\pi} \right) \times \cos(X_i^g), \quad (11)$$

$$S_2^g = R^g \times \left(\frac{rand \times X_i^g}{2\pi} \right) \times \sin(X_i^g). \quad (12)$$

where the vulture's next updated position at the next $(g + 1)^{th}$ iteration is denoted by X_i^{g+1} , D_i^g is defined using Equation (8), and $rand$ is an arbitrary value between 0 and 1. The distance d_i^g between the vulture and one of the best two vultures is estimated using Equation (10); R^g denotes the appropriate best vulture in the current g^{th} iteration, which is computed using Equation (2); S_1^g and S_2^g are computed employing Equations (11) and (12), respectively; and X_i^g denotes the current position at the g^{th} iteration.

- **Second sub-phase of exploitation:** This sub-phase is carried out when $|F_i^g|$ is less than 0.5. Various vulture species gather around the food supply and engage in many sieges and brawls during this sub-phase. This sub-phase employs two different strategies. A predefined parameter P_3 and an arbitrary value $rand_{P_3}$, with values ranging from 0 to 1, are used to decide which of these two strategies to use.

This sub-phase's initial strategy is called **assemble vultures around the food source**. In this strategy, different kinds of vultures search for food and may compete near a single source. The second strategy is known as **hostile siege fight**. In this strategy, the vultures become more aggressive and try to plunder the leftover food from the healthy vultures by flocking around them in different ways. The healthy vultures, on the other hand, deteriorate and are unable to fend off the other vultures. The following illustrates the strategies for the second exploitation sub-phase:

$$X_i^{g+1} = \begin{cases} \frac{A_1^g + A_2^g}{2}, & \text{if } rand_{P_3} \leq P_3, \\ R^g - |d_i^g| \times F_i^g \times Levy_d, & \text{if } rand_{P_3} > P_3, \end{cases} \text{ if } |F_i^g| < 0.5, \quad (13)$$

$$A_1^g = BestVulture_1^g - \frac{BestVulture_1^g \times X_i^g}{BestVulture_1^g - (X_i^g)^2} \times F_i^g, \quad (14)$$

$$A_2^g = SecondBestVulture_2^g - \frac{SecondBestVulture_2^g \times X_i^g}{SecondBestVulture_2^g - (X_i^g)^2} \times F_i^g, \quad (15)$$

$$Levy_d = 0.01 \times \frac{\mu \times \sigma}{|v|^{\frac{1}{\beta}}}, \quad (16)$$

$$sigma = \left(\frac{\Gamma(1 + \beta) \times \sin(\frac{\Pi\beta}{2})}{\Gamma(1 + \beta/2) \times \beta \times 2^{(\frac{\beta-1}{2})}} \right)^{\frac{1}{\beta}}. \quad (17)$$

where the next updated position of the vulture at the next $(g + 1)^{th}$ iteration, reflecting the assembly of vultures, is denoted by X_i^{g+1} . A_1^g and A_2^g are evaluated using, respectively, Equations (14) and (15). To increase the effectiveness of the AVO algorithm, $Levy_d$ is the levy flight distribution function obtained using Equation (16). d is the dimensional space; σ is defined by Equation (17), where $\beta = 1.5$ is a constant value; and μ and v are random numbers distributed equally within the range $[0, 1]$.

3.4. Converting to Binary Nature

In the presented AVO algorithm, the positions are shown as real values. As such, they are not immediately applicable to the GS binary issue. Therefore, there is a need to convert these real position values into binary values to conform to GS's binary nature and maintain the original algorithm's structure. In this conversion, 1s represent the real values of the pertinent selected genes in the binarization vector, whereas 0s represent the real values of the unselected genes, which are not pertinent. At each iteration g , the following mathematical expression can be used to convert the real position X_i^g to a binary position $(X_i^g)_{bin}$:

$$(X_i^g)_{bin} = \begin{cases} 1 & \text{if } X_i^g > \delta, \\ 0 & \text{otherwise.} \end{cases} \quad (18)$$

where an arbitrary threshold point within $[0, 1]$ is represented by δ . According to this fundamental binary conversion approach, the binary “1” (selected feature) replaces its real value if $(X_i^g)_{bin}$ is greater than δ . On the other hand, if $(X_i^g)_{bin}$ is smaller than δ , its real value is set to the binary “0” (a feature that was not chosen).

3.5. Appraising the Fitness Function Value

Finding the fewest number of selected features and optimizing the classification accuracy of the available classifiers (k -NN and SVM models) are two conflicting objectives that should be balanced to achieve the best solution and determine its quality. Since the k -NN and SVM classifiers’ accuracies might be hampered if the number of selected features is lower than the optimal, the fitness function balances the selected features’ size and accuracy. The fitness function concentrates on lowering the classification error rate rather than accuracy, as follows:

$$fit = w_1 \times Err_{rate} + w_2 \times \frac{|feat_{picked}|}{|D|}, \quad (19)$$

$$w_1 \in [0, 1], w_2 = 1 - w_1.$$

where $feat_{picked}$ denotes the number of selected features, D denotes the total number of features in the dataset, and Err_{rate} represents the classification error rate from the k -NN and SVM classifiers. The classification accuracy importance and the number of selected features are denoted by the weight parameters w_1 and w_2 , respectively. The values of w_1 (0.99) and w_2 (0.01) are determined through extensive trials conducted in previous studies [21,42,43].

3.6. Incorporating the DE Technique

DE [44] is characterized by its high efficiency and simplicity in finding a suitable solution for complex optimization problems. It can quickly produce value-added results. Three main processes—mutation, crossover, and selection—are necessary for DE. The differential mutation process seeks to produce a modified vector v_i for each iteration’s solution vector. This can be computed mathematically as follows:

$$\vec{v}_i = \vec{X}_{r_1} + W_M(\vec{X}_{r_2} - \vec{X}_{r_3}) \quad (20)$$

Within the range $[1, \text{population size}]$, three randomly selected asymmetric vectors are denoted by the symbols X_{r_1} , X_{r_2} , and X_{r_3} . W_M denotes the mutation weighting factor within the interval $[0, 1]$.

After the mutation operation, DE performs a crossover operation to increase the population’s variety. An offspring vector u_i is produced by combining values from the target vector X_i and the altered vector v_i . The most commonly used and basic factor of crossover search is binary crossover, which has the following mathematical expression:

$$u_{i,d} = \begin{cases} v_{i,d}, & \text{if } rand \leq C_R \text{ or } d = j_{rand}, \\ X_{i,d}, & \text{otherwise.} \end{cases} \quad (21)$$

where a uniformly distributed random number $j_{rand} \in [1, 2, \dots, D_X]$ is used to ensure that the mutated vector has at least one dimension. The possibility of crossing each element is calculated using the crossover rate C_R , frequently determined to be a big amount ($C_R = 0.9$).

The selection operation is then carried out, as shown in Equation (22). Here, a comparison between the fitness function $f(X_i)$ of the target vector and the corresponding offspring vector $f(u_i)$ is performed, and the lowest-valued fitness function is kept for the upcoming iteration.

$$X_i = \begin{cases} u_i, & \text{if } f(u_i) < f(X_i) \\ X_i, & \text{otherwise.} \end{cases} \quad (22)$$

3.7. The Complete RBAVO-DE Algorithm

To handle the GS strategy, the steps of the recommended RBAVO-DE algorithm are described in the following subsections. The pseudo-code for the proposed RBAVO-DE algorithm is provided in Algorithm 1. A flowchart of the proposed RBAVO-DE algorithm is shown in Figure 1, illustrating its main steps.

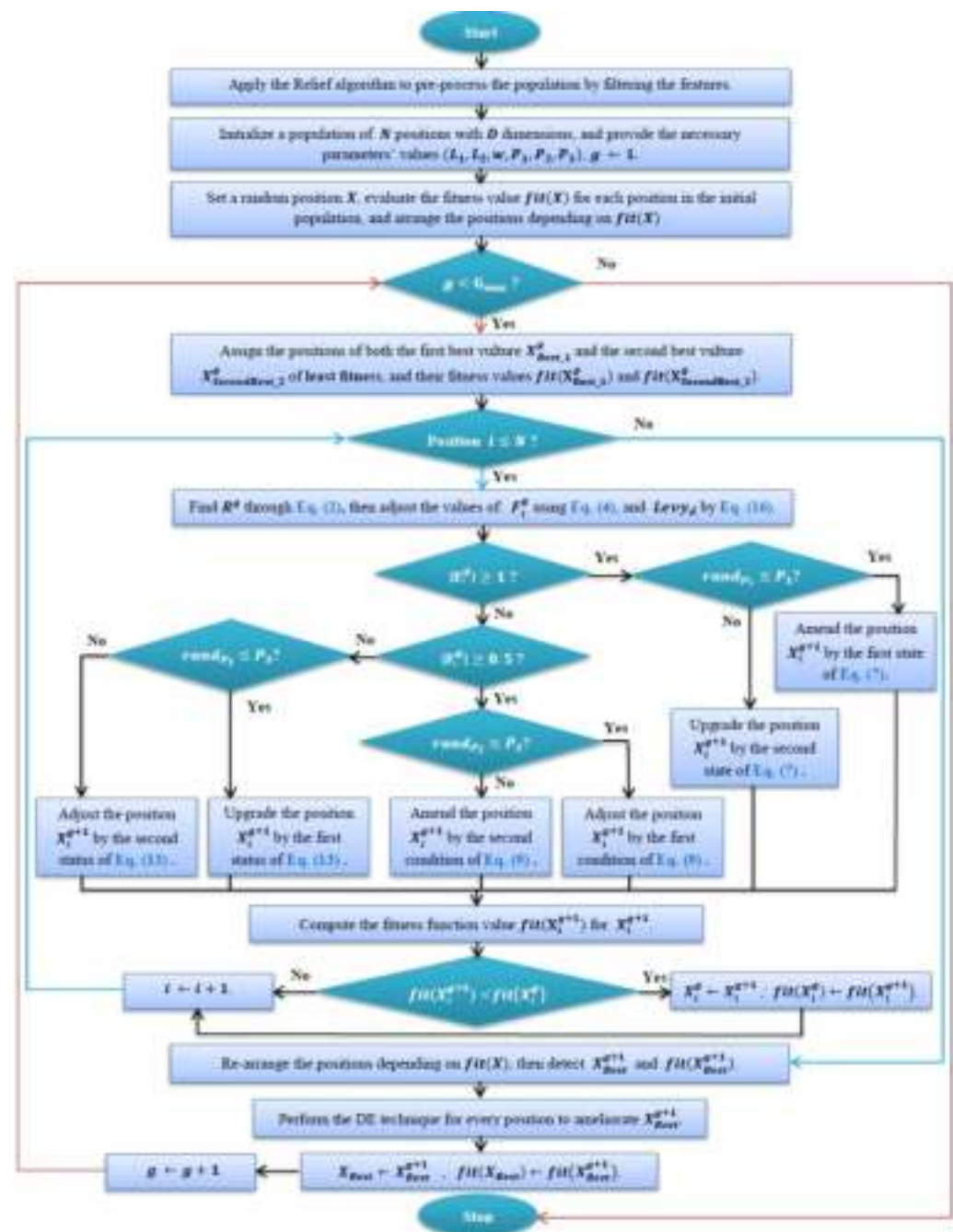


Figure 1. Flowchart of the RBAVO-DE algorithm.

Algorithm 1 The proposed RBAVO-DE algorithm.**Input:**

N —total number of positions (size of population)
 G_{max} —maximum number of permitted iterations
 D —problem's dimensional space
 LB —lower bounds of variables
 UB —upper bounds of variables
 C_R —crossover rate
 W_M —mutation weighting factor

Output:

X_{Best} —the global best vulture's position found while searching
 $fit(X_{Best})$ —the global best fitness function value found, which should be lessened

```

1: Start
2:   Apply the Relief approach for selecting the related features and filtering them, as demonstrated in Section 3.1;
3:   Initialize a population of  $N$  positions, and provide the values of the necessary parameters ( $L_1$ ,  $L_2$ ,  $w$ ,  $P_1$ ,  $P_2$ , and  $P_3$ );
4:   Set a random position  $X$  in the initial population;
5:   Evaluate the fitness values  $fit(X)$  for each position in the initial population;
6:   Arrange the positions in ascending order depending on their fitness function  $fit(X)$ ;
7:    $g \leftarrow 1$ ; ▷ Current number of iterations
8:   while  $g < G_{max}$  do
9:     Assign the positions of both the first-best vulture  $X_{Best_1}^g$  and the second-best vulture  $X_{SecondBest_2}^g$ , as well as their fitness
       values  $fit(X_{Best_1}^g)$  and  $fit(X_{SecondBest_2}^g)$ , among all positions in the population;
10:    for vulture's position  $i = 1 : N$  do
11:      Find the best vulture  $R^g$  using Equation (2);
12:      Adjust the vulture's famine level  $F_i^g$  using Equation (4);
13:      Modify the levy flight distribution function  $Levy_d$  using Equation (16);
14:      if  $|F_i^g| \geq 1$  then
15:        if  $rand_{p_1} \leq P_1$  then
16:          Amend the vulture's position  $X_i^{g+1}$  based on the first stage of the exploration phase using Equation (7);
17:        else if  $rand_{p_1} > P_1$  then
18:          Upgrade the vulture's position  $X_i^{g+1}$  based on the second stage of the exploration phase using Equation (7);
19:        end if
20:      else if  $|F_i^g| < 1$  then
21:        if  $|F_i^g| \geq 0.5$  then
22:          if  $rand_{p_2} \leq P_2$  then
23:            Adjust the vulture's position  $X_i^{g+1}$  based on the first condition of the first exploitation sub-phase using
24:            Equation (9);
25:          else if  $rand_{p_2} > P_2$  then
26:            Amend the vulture's position  $X_i^{g+1}$  based on the second condition of the first exploitation sub-phase
27:            using Equation (9);
28:          end if
29:        else if  $|F_i^g| < 0.5$  then
30:          if  $rand_{p_3} \leq P_3$  then
31:            Upgrade the vulture's position  $X_i^{g+1}$  based on the first status of the second exploitation sub-phase using
32:            Equation (13);
33:          else if  $rand_{p_3} > P_3$  then
34:            Adjust the vulture's position  $X_i^{g+1}$  based on the second status of the second exploitation sub-phase using
35:            Equation (13);
36:          end if
37:        end if
38:      end if
39:       $fit(X_i^{g+1}) \leftarrow$  Compute the fitness function value for  $X_i^{g+1}$ ;
40:      if  $fit(X_i^{g+1}) < fit(X_i^g)$  then
41:         $X_i^g \leftarrow X_i^{g+1}$ ;
42:         $fit(X_i^g) \leftarrow fit(X_i^{g+1})$ ;
43:      end if
44:    end for
45:    Re-arrange the positions in ascending order depending on their fitness function  $fit(X)$ ;
46:    Detect the global best position  $X_{Best}^{g+1}$  and its global best fitness value  $fit(X_{Best}^{g+1})$  when the current iteration  $g + 1$  is over;
47:    Perform the DE technique for every position to ameliorate  $X_{Best}^{g+1}$ , as shown in Section 3.6;
48:     $X_{Best} \leftarrow X_{Best}^{g+1}$ ;
49:     $fit(X_{Best}) \leftarrow fit(X_{Best}^{g+1})$ ;
50:     $g \leftarrow g + 1$ ;
51:  end while
52: End

```

4. Experimental Results and Discussion

This section presents the empirical findings of the proposed RBAVO-DE and its counterparts: Binary Artificial Bee Colony (BABC) [45], Binary Salp Swarm Algorithm (BSSA) [46], Binary PSO (BPSO) [47], Binary Bat Algorithm (BBA) [48], Binary Grey-Wolf Optimization (BGWO) [49], Binary Grasshopper Optimization Algorithm (BGOA) [50], Binary Whale Optimization Algorithm (BWOA) [51], Binary ASO (BASO) [52], Binary Bird Swarm Algorithm (BBSA) [53], Binary HGSO (BHGSO) [54], and Binary Harris Hawks Op-

timization (BHHO) [55]. The optimizers undergo evaluation through training and testing benchmarks, with conclusive results derived from the average values of the evaluation metrics. The benchmarks employed for assessing the performance of the proposed model are detailed in Section 4.1. The parameters used in the operational environments are outlined in Section 4.2. The metrics used for evaluation are described in Section 4.3. The analysis of the experimental outcomes is discussed in Section 4.4.

4.1. Dataset Description

In-depth experimental approaches and various wrapper algorithms were applied to twenty-two datasets of gene descriptions; the data comprise normalized Level 3 RNA-Seq gene expression data for twenty-two kinds of tumors from the Broad Institute. These data are publicly accessible and can be found in [56]. We adhered to the methodology described in [36] and observed discrepancies between the data referenced from GitHub in the paper and the figures reported within the document, which were sourced from the website. The website's data included a mix of tumor and standard samples, whereas the paper treated the data uniformly as tumor samples. Consequently, we conducted a detailed examination of the data. Initially, the website offered various formats of the identical dataset we intended to analyze. Upon delving into the data, we encountered the following challenges:

- Certain genes are identified by ID but lack an associated symbol.
- Some genes are absent from the annotation file.
- There is a mix-up of samples, including both normal and tumor types.

Consequently, pre-processing was necessary to segregate and identify samples, distinguish between normal and tumor samples for use in binary classification, and streamline the GS process. We addressed the challenges above in the following manner:

- We looked up the corresponding gene symbol for each ID found in the annotation file.
- After cross-referencing with the annotating file, over one hundred genes were eliminated.
- Based on the sample report, every Excel sheet's row was organized according to the kind of sample for binary classification.

Moreover, the Relief approach, as detailed in Section 3.1, was utilized in the pre-processing stage to calculate the weight of each gene within the benchmark. These weights were subsequently ordered from highest to lowest. Genes with lower weights were then removed. The Relief approach was useful for discarding genes that do not contribute to classification.

Following the pre-processing phase, the data were refined and ready for the GS process. Contrary to the approach used in [36], which tackled the multi-classification of all types of cancer together, we opted to analyze every cancer type individually for greater specificity. Table 1 [57] presents a comprehensive overview of all twenty-two tumor types and their respective sample counts. The number of features utilized in the benchmark datasets is 32,000 features (genes).

Table 1. Datasets employed in this paper [57].

#	Cancer	Normal Records	Tumor Records
1	Bladder Urothelial Carcinoma (BLCA)	19	408
2	Thyroid Cancer (THCA)	59	501
3	Cervical and Endocervical Cancers (CESCs)	3	304
4	Cholangiocarcinoma (CHOL)	9	36
5	Colon Adenocarcinoma (COAD)	41	458
6	Esophageal Cancer (ESCA)	13	184
7	Glioblastoma Multiforme (GBM)	5	153

Table 1. *Cont.*

#	Cancer	Normal Records	Tumor Records
8	Thymoma (THYM)	2	120
9	Head and Neck Squamous Cell Carcinoma (HNSC)	44	520
10	Kidney Chromophobe (KICH)	25	66
11	KIRC	72	533
12	Kidney Renal Papillary (KIRP)	32	290
13	Liver Hepatocellular Carcinoma (LIHC)	50	371
14	LUAD	59	515
15	LUSC	51	501
16	Pancreatic Adenocarcinoma (PAAD)	4	178
17	UCEC	35	176
18	Pheochromocytoma and Paraganglioma (PCPG)	3	179
19	Rectum adenocarcinoma (READ)	10	94
20	Sarcoma (SARC)	2	259
21	Skin Cutaneous Melanoma (SKCM)	1	103
22	STAD	37	415

4.2. Parameter Setting

The proposed RBAVO-DE algorithm was compared against binary variants of different meta-heuristic optimizers, such as BABC, BSSA, BPSO, BBA, BGWO, BWOA, BBSA, BGOA, BHHO, BASO, and BHGSO. The critical parameters for the ML models employed in this study are detailed in Table 2.

Table 2. The key parameters of the employed ML models.

Model	Parameters
SVM	Polynomial kernel = 2
k-NN	Euclidean distance metric $k = 5$ [21,58,59]

Within the proposed framework, the validity of the results was verified using a 10-fold cross-validation approach to ensure the reliability of the outcomes. This involved randomly dividing each dataset into two distinct subsets: 80% for training and the remaining 20% for testing. The training portion was employed to train the presented classifiers using optimization techniques, while the testing portion was used to evaluate the selected genes' effectiveness. The parameters most commonly applied across all compared algorithms are summarized in Table 3. To execute all experiments in this paper, Python was utilized in a computational environment with a Dual Intel® Xeon® Gold 5115 2.4 GHz CPU and 128 GB of RAM on the Microsoft Windows Server 2019 operating system.

Table 3. Parameter configurations of all utilized optimizers.

Optimizer	Parameters
All optimizers	Execution count = 30 Maximum count of iterations $G_{max} = 100$ Size of population $N = 10$

Table 3. Cont.

Optimizer	Parameters
Proposed RBAVO-DE	Dimensionality D = The count of genes in the employed datasets Upper boundaries UB Lower boundaries LB Parameter $L_1 = 0.7$ Parameter $L_2 = 0.2$ Parameter $w = 2$ Parameter $P_1 = 0.6$ Parameter $P_2 = 0.6$ Parameter $P_3 = 0.5$
BSSA	Count of scroungers $SD = 0.1 \times N$ Count of producers $PD = 0.2 \times N$ Safety threshold $ST = 0.8$
BABC	Count of employed bees = 16 Count of scout bees = 3 Count of onlooker bees = 4
BPSO	Inertia weight ($\omega_{max} = 0.9\omega_{min} = 0.4$) Acceleration coefficients ($c_2 = c_1 = 1.2$)
BBA	Loudness $A = 0.8$ Pulse emission rate $r = 0.95$
BWOA	Lower and upper pulse frequencies = 0, 10 a is linearly decreased from 2 to 0 $p = 0.5$ $b = 1.0$
BHHO	Rabbit energy $E \in [-1, 1]$
BGWO	a is linearly decreased from 2 to 0
BGOA	$C_{min} = 0.00004$ and $C_{max} = 1$
BBSA	Frequency of flight $ff = 10$ Effect on birds' vigilance Followed coefficient $fl = 0.5$ Probability of foraging for food $p = 0.8$ Acceleration coefficients ($c_1 = c_2 = 1.5$) behaviors ($a_1 = a_2 = 1.0$)
BASO	Depth weight $\alpha = 50$ Multiplier weight $\beta = 0.2$
BHGSO	Count of clusters = 2 $\alpha = \beta = 0.1$ $K = 1.0$, and $l_3 = 1E - 02$ $l_1 = 5E - 03$, $l_2 = 1E + 02$

4.3. Evaluation Criteria

To evaluate the proposed RBAVO-DE's effectiveness relative to other methods, each strategy was independently tested thirty times on each benchmark to ensure statistical validation of the results. For this purpose, the following established performance metrics for the GS issues were employed:

- **Mean classification accuracy** ($Mean_{AC}$): This measure represents the accuracy of correctly classifying data, calculated by running the algorithm independently thirty times. It is determined in the following way:

$$Mean_{AC} = \sum_{k=1}^{30} \sum_{r=1}^m match(PL_r, AL_r) \quad (23)$$

In this formula, m denotes the number of samples in the test dataset, while PL_r and AL_r denote the predicted labels from the classifier and the actual class labels for sample r , respectively. The function $match(PL_r, AL_r)$ serves as a comparison mechanism, where if PL_r is equal to AL_r , then $match(PL_r, AL_r)$ is assigned a value of 1; if they do not match, it is assigned a value of 0.

- **Mean fitness** ($Mean_{Fi}$): This measure calculates the average $meanFi$, achieved by running the proposed approach thirty times independently. It gauges the balance between lowering the classification error rate and minimizing the number of genes selected. A lower value indicates a superior solution, assessed based on fitness:

$$MeanFi = \frac{1}{30} \sum_{k=1}^{30} f_*^k \quad (24)$$

where f_*^k shows the optimum fitness value achieved in the k – th execution.

- **Mean number of chosen genes** ($MeanGe$): This metric calculates the average number of genes chosen (or GS ratio) by running the proposed approach thirty times independently, and it is defined as follows:

$$MeanGe = \frac{1}{30} \sum_{k=1}^{30} \frac{|d_*^k|}{|D|} \quad (25)$$

Here, $|d_*^k|$ denotes the total count of genes chosen in the optimal solution for the k^{th} execution, and $|D|$ represents the total number of genes present in the initial benchmark.

- **Standard Deviation** ($STDE$): Based on the outcomes above, the overall average results derived from thirty separate executions of each optimizer on every benchmark were assessed for stability in the following manner:

$$STDE = \sqrt{\frac{1}{29} \sum_{k=1}^{30} (Y_*^k - Mean_Y)^2} \quad (26)$$

where Y shows the measure to be utilized, Y_*^k denotes the measure value Y in the k^{th} run, and $Mean_Y$ is the mean of the measure from thirty independent executions.

The data in the subsequent tables represent the mean values obtained from thirty independent executions, focusing on classification accuracy, the number of chosen genes, average fitness, precision, recall, and F1-score. The ensuing subsections thoroughly examine and discuss these experimental findings, with bold figures highlighting the optimal results.

4.4. Results of Comparing the Proposed RBAVO-DE with Various Meta-Heuristic Algorithms

The proposed RBAVO-DE approach with the SVM and k -NN models was compared with other recent meta-heuristic methods executed under the same conditions to show its superiority over its counterparts. The proposed RBAVO-DE algorithm was compared with binary variants of several optimizers, including BABC, BSSA, BPSO, BBA, BGWO, BWOA, BBSA, BGOA, BHHO, BASO, and BHGSO.

4.4.1. Results Employing the k -NN Model

Table 4 shows the results of the proposed RBAVO-DE algorithm and other optimization techniques employing the k -NN model, based on accuracy metrics compared under similar conditions. The experimental outcomes reveal that the proposed RBAVO-DE achieved the most promising outcomes in four benchmarks. It is noteworthy that all competitive methods, including the proposed RBAVO-DE employing the k -NN model, produced comparable outcomes across eighteen benchmarks.

The rest of the metrics for the k -NN model are shown in Appendix A.1. In terms of the average fitness values, the proposed RBAVO-DE algorithm demonstrated higher efficiency compared to its counterparts using the k -NN model under similar conditions. The RBAVO-DE algorithm yielded the lowest fitness results and the most competitive STDE across all benchmarks. Also, all utilized benchmarks are high-dimensional, demonstrating that the proposed RBAVO-DE can run effectively on all benchmarks regardless of size. The RBAVO-DE algorithm shows promise by effectively balancing exploitation and exploration of the search space, avoiding becoming trapped in local optima during iterations. Unlike

Regarding the mean results for the genes selected by the proposed RBAVO-DE algorithm and its counterparts employing k -NN, the proposed RBAVO-DE algorithm surpassed the other methods across all benchmarks concerning the number of chosen genes. Furthermore, the RBAVO-DE's capacity to identify significant genes is due to its effective exploration of possible areas while enhancing accuracy.

Regarding the mean recall of the proposed RBAVO-DE and its counterparts employing k -NN, RBAVO-DE outperformed the alternative approaches for three of the twenty-two datasets. On the other hand, BSSA, BABC, BPSO, BGWO, BWOA, BGOA, and BBBSA produced the same outcomes as RBAVO-DE for 19 datasets, while BHHSO achieved similar results for 18 datasets. BHGSO, which obtained the same outcomes as the proposed RBAVO-DE for seventeen datasets, ranked fourth. Ultimately, BASO and BBA ranked lowest among all approaches, producing similar results to RBAVO-DE for sixteen datasets.

With regard to the mean F1-score of the proposed RBAVO-DE and its counterparts employing k -NN, the proposed RBAVO-DE outperformed the alternative approaches for three of the twenty-two datasets. In contrast, for eighteen datasets, BSSA, BABC, BPSO, BGWO, BGOA, and BBSA produced the same outcomes as RBAVO-DE, while BWOA and BHHO produced comparable outcomes for seventeen datasets. BASO and BHGSO, which obtained the same outcomes for 16 datasets as the proposed RBAVO-DE, ranked fourth. Ultimately, BBA ranked lowest among all approaches, producing similar outcomes to the proposed RBAVO-DE for fourteen datasets.

[illegible]

Table 4. Cont.

Dataset	Metric	RBAVO-DE	BSSA	BABC	BPSO	BBA	BGWO	BWOA	BGOA	BHHO	BBSA	BASO	BHGSO
PCPG	Average	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
	STDE	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
READ	Average	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
	STDE	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
SARC	Average	0.981	0.981	0.981	0.981	0.981	0.981	0.981	0.981	0.981	0.981	0.981	0.981
	STDE	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
SKCM	Average	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
	STDE	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
STAD	Average	0.999	0.996	0.999	0.996	0.989	0.996	0.995	0.997	0.992	0.995	0.996	0.989
	STDE	0.003	0.005	0.004	0.005	0.000	0.005	0.006	0.005	0.005	0.006	0.005	0.000
THCA	Average	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
	STDE	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
THYM	Average	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
	STDE	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
UCEC	Average	0.998	0.985	0.988	0.977	0.969	0.984	0.977	0.978	0.978	0.979	0.976	0.968
	STDE	0.008	0.012	0.013	0.006	0.011	0.012	0.006	0.008	0.009	0.009	0.005	0.011
Ranking	W T L	4 18 0	0 18 4	0 18 4	0 18 4	0 18 4	0 18 4	0 18 4	0 18 4	0 18 4	0 18 4	0 18 4	0 18 4

Boldface values denote the best results.

4.4.2. Results Employing the SVM Model

Table 5 displays the outcomes of the proposed RBAVO-DE and various optimization techniques employing the SVM model concerning classification accuracy results assessed under identical running conditions. The experimental outcomes demonstrate that the proposed RBAVO-DE algorithm outperformed the other approaches by obtaining the most promising values for four datasets. All competitive methods, including the proposed RBAVO-DE employing SVM, obtained equivalent values across 18 benchmarks.

The rest of the metrics utilizing SVM are presented in Appendix A.2. Regarding the mean fitness values of the proposed RBAVO-DE and its counterparts employing the SVM model under equivalent running conditions, the proposed RBAVO-DE proved to be more efficient than the other methods. The proposed RBAVO-DE employing the SVM model yielded the lowest fitness values and the most competitive STDE across all benchmarks. Also, all utilized benchmarks are high-dimensional, demonstrating that the proposed RBAVO-DE can run effectively on all benchmarks regardless of size. The RBAVO-DE algorithm shows promise by effectively balancing exploitation and exploration of the search space, avoiding becoming trapped in local optima during iterations. Unlike many other algorithms that tend to become trapped, it demonstrated the ability to escape such traps.

Regarding the mean results for the genes selected by the proposed RBAVO-DE and its counterparts employing SVM, the proposed RBAVO-DE achieved more promising results than other techniques across all benchmarks utilized in this study. Also, the superiority of the proposed RBAVO-DE employing SVM in this context demonstrates its capability to effectively explore valuable regions of the search space while avoiding regions with non-feasible solutions.

With regard to the average precision values of the proposed RBAVO-DE and its counterparts employing SVM, the proposed RBAVO-DE outperformed the alternative approaches for three of the twenty-two datasets. On the other hand, for eighteen datasets, BABC, BGWO, BWOA, BBSA, and BASO yielded comparable results. For seventeen datasets, BSSA, BPSO, BGOA, and BHHO yielded results similar to those of RBAVO-DE, thereby ranking fourth. Ultimately, BBA ranked lowest among all approaches, producing similar outcomes to those of the proposed RBAVO-DE for twelve datasets.

In terms of the mean recall of the proposed RBAVO-DE method and its counterparts using SVM, the proposed RBAVO-DE outperformed the alternative approaches for three of the twenty-two datasets. In contrast, BPSO and BHHO performed similarly for eighteen datasets, while BSSA, BABC, BGWO, BWOA, BGOA, and BBSA produced the same results as the proposed RBAVO-DE for nineteen datasets. BHGSO, which obtained the same outcomes as RBAVO-DE for seventeen datasets, ranked fourth. In the end, BASO and BBA ranked lowest among all approaches, producing outcomes similar to those of RBAVO-DE for sixteen datasets.

Regarding the mean F1-score of the proposed RBAVO-DE algorithm and its counterparts employing SVM, the proposed RBAVO-DE outperformed the alternative approaches for five of the twenty-two datasets. On the other hand, BWOA produced outcomes similar to those of the proposed RBAVO-DE for seventeen datasets, while BSSA, BABC, BPSO, BGWO, BGOA, and BBSA produced comparable results for sixteen datasets. BHHO, which obtained the same outcomes as the proposed RBAVO-DE for fifteen datasets, ranked fourth. Ultimately, BBA ranked lowest among all approaches, producing similar outcomes to those of RBAVO-DE for 12 datasets.

Table 5. The results of the proposed RBAVO-DE algorithm and its counterparts employing SVM regarding average classification accuracy values.

Dataset	Metric	RBAVO-DE	BSSA	BABC	BPSO	BBA	BGWO	BWOA	BGOA	BHHO	BBSA	BASO	BHGSO
BLCA	Average	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
	STDE	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
CESC	Average	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
	STDE	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
CHOL	Average	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
	STDE	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
COAD	Average	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
	STDE	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
ESCA	Average	0.986	0.975	0.977	0.974	0.974	0.977	0.974	0.975	0.975	0.975	0.974	0.974
	STDE	0.013	0.004	0.008	0.000	0.000	0.008	0.000	0.005	0.005	0.005	0.000	0.000
GBM	Average	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
	STDE	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
HNSC	Average	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
	STDE	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
KICH	Average	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
	STDE	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
KIRC	Average	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
	STDE	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
KIRP	Average	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
	STDE	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
LIHC	Average	1.000	1.000	1.000	1.000	0.9992	1.000	1.000	1.000	1.000	1.000	1.000	1.000
	STDE	0.000	0.000	0.000	0.000	0.0029	0.000	0.000	0.000	0.000	0.000	0.000	0.000
LUAD	Average	0.993	0.991	0.991	0.991	0.991	0.991	0.991	0.991	0.991	0.991	0.992	0.991
	STDE	0.003	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.002	0.000
LUSC	Average	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
	STDE	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
PAAD	Average	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
	STDE	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
PCPG	Average	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
	STDE	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
READ	Average	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
	STDE	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
SARC	Average	0.994	0.986	0.989	0.986	0.981	0.989	0.987	0.986	0.984	0.986	0.984	0.981
	STDE	0.008	0.008	0.009	0.008	0.000	0.009	0.009	0.008	0.006	0.008	0.007	0.000
SKCM	Average	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
	STDE	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
STAD	Average	0.990	0.989	0.989	0.989	0.989	0.989	0.989	0.989	0.989	0.989	0.989	0.989
	STDE	0.003	0.000	0.000	0.000	0.002	0.000	0.000	0.000	0.002	0.000	0.000	0.000
THCA	Average	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
	STDE	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
THYM	Average	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
	STDE	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
UCEC	Average	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
	STDE	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Ranking	W T L	4 18 0	0 18 4	0 18 4	0 18 4	0 18 4	0 18 4	0 18 4	0 18 4	0 18 4	0 18 4	0 18 4	0 18 4

Boldface numbers indicate the best results.

4.5. Convergence Analysis

It is evident in Appendix B [57] that the proposed RBAVO-DE employing the SVM and k -NN models attained optimum convergence behavior across all benchmarks. Therefore, the convergence performance of the proposed RBAVO-DE employing the SVM and k -NN models demonstrates its capability to reach optimal outcomes promptly while maintaining an efficient equilibrium between search and exploitation.

4.6. Wilcoxon's Rank-Sum Test

This paper compares the fitness values achieved by the proposed RBAVO-DE and its counterparts pairwise using the Wilcoxon rank-sum test [60], which aims to determine whether there is a statistically significant difference between the different approaches. This test is a crucial tool for evaluating the success of the proposed algorithm. The Wilcoxon test is employed in hypothesis testing to compare matched data. It involves sorting the absolute differences in the outcomes of paired procedures on the j^{th} of N issues. Next, the lowest value is found by summing the positive (R^+) and negative (R^-) rankings. The null hypothesis is rejected if the resultant significance level is less than 5%, and not rejected otherwise.

From examining the results of the comparison made between the RBAVO-DE and its counterparts pairwise using the Wilcoxon test, we found that all R^+ values equal 253, all R^- values equal 0, and all p -values equal 4.768×10^{-7} which are less than 0.05 (significance level of 5%). Hence, it can be concluded that the proposed RBAVO-DE employing SVM and k -NN performs better than any other algorithm in all cases. Therefore, all p -values less than 0.05 (significance level of 5%) offer compelling evidence that the outcomes of the proposed strategy are statistically significant and not just coincidental.

4.7. Computational Complexity of the RBAVO-DE Algorithm and Various Meta-Heuristic Algorithms

4.7.1. Computational Complexity of the Execution Time of the RBAVO-DE Algorithm

Each of RBAVO-DE's five core steps can be analyzed separately to determine its computational complexity. These steps include filtering features, initializing the population, boosting and amending the position, appraising the fitness function, and using DE. Then, $O_{execute\_time}(RBAVO - DE)$ can be utilized to represent the overall computational complexity of the proposed RBAVO-DE algorithm. This can be computed using the following big-O notation formulas:

$$\begin{aligned} O_{execute\_time}(RBAVO - DE) = & O_{execute\_time}(\text{Filtering features}) + \\ & O_{execute\_time}(\text{Initializing population}) + \\ & O_{execute\_time}(\text{Boosting and amending position}) + \\ & O_{execute\_time}(\text{Appraising fitness function}) + \\ & O_{execute\_time}(\text{Incorporating DE technique}). \end{aligned} \quad (27)$$

Since N determines the population size, G_{max} is the maximum number of iterations allowed, and D is the problem's dimensional space. Hence,

$$\begin{aligned} O_{execute\_time}(\text{Filtering features}) &= O_{execute\_time}(D). \\ O_{execute\_time}(\text{Initializing population}) &= O_{execute\_time}(N). \\ O_{execute\_time}(\text{Boosting and amending position}) &= O_{execute\_time}(G_{max} \times N \times D). \\ O_{execute\_time}(\text{Appraising fitness function}) &= O_{execute\_time}(G_{max} \times N). \\ O_{execute\_time}(\text{Incorporating DE technique}) &= O_{execute\_time}(N \times D). \end{aligned}$$

For that,

$$\begin{aligned} O_{execute\_time}(RBAVO - DE) &= O_{execute\_time}(D) + \\ & O_{execute\_time}(N) + O_{execute\_time}(G_{max} \times N \times D) + \\ & O_{execute\_time}(G_{max} \times N) + O_{execute\_time}(N \times D) = \\ & O_{execute\_time}(G_{max} \times N \times D). \end{aligned}$$

4.7.2. Computational Complexity of the Memory Usage of the RBAVO-DE Algorithm

This involves measuring the amount of memory required by an algorithm to tackle a problem as the quantity of the input increases. It is frequently expressed as the additional memory needed by the algorithm beyond the input. It entails merging the following two primary elements:

1. **Memory usage of input variables:** This is the amount of memory required for the algorithm to store the input data. There are 13 input variables related to the proposed RBAVO-DE algorithm, as follows: N , G_{max} , D , LB , UB , C_R , W_M , L_1 , L_2 , w , P_1 , P_2 , and P_3 . Since each variable stores numerical values, 4 bytes of memory are utilized by each one. Consequently, the total memory usage complexity of these 13 input variables is 52 bytes ($13 \times 4 \text{ bytes} = 52 \text{ bytes}$). The memory usage complexity of the input values is constant.
2. **Additional memory usage:** This shows how much more memory the algorithm needs in addition to the input. It includes the memory required for data structures, internal variables, and other components of the algorithm. Regardless of the size of the input, the RBAVO-DE algorithm requires a specific amount of extra memory. The following variables are involved:
 - The memory usage complexity consumed by $X_{initial}$ is $(4 \times N \times D)$ bytes. This is because each position in the positions vector $X_{initial}$ requires 4 bytes of memory, and its size is $(N \times D)$, proportionate to the initial population of N positions with dimension size D . Because the amount of memory required rises linearly with the value $(N \times D)$, its memory use complexity is linear.
 - The variables pr , g , F_i , t , R , D_i , d_t , S_1 , S_2 , A_1 , A_2 , σ , $fit(X_i)$, $fit(X_{Best_1})$, $fit(X_{SecondBest_2})$, and $fit(X_{Best})$ require 4 bytes of memory space each since they only represent numerical values. As a result, the total memory usage complexity of these 16 variables is 64 bytes ($16 \times 4 \text{ bytes} = 64 \text{ bytes}$). This memory usage complexity is constant.
 - The position vectors $BestVulture_1$, $SecondBestVulture_2$, X_i , $Levy_d$, μ , σ , X_i^{bin} , v_i , X_{r_1} , X_{r_2} , X_{r_3} , u_i , and X_{Best} each require 4 bytes of memory space, and the size of each vector is D , proportionate to the dimension size of the acquired positions. As a result, the vectors at these 13 points have a total memory usage complexity of $(52 \times D)$ bytes ($13 \times 4 \times D \text{ bytes}$). Since the memory needed grows linearly with the value D , its memory consumption complexity is linear.

As a result, the overall memory usage complexity for all of the additional variables listed above is $(4 \times N \times D) + 64 + (52 \times D)$ bytes.

Lastly, the following formula can be used to determine the computational complexity of the overall memory usage of the proposed RBAVO-DE algorithm:

$$\begin{aligned} \text{Memory usage complexity(RBAVO-DE)} &= \text{Input values memory usage} + \\ &\text{Additional memory usage} = 52 + ((4 \times N \times D) + 64 + (52 \times D)) \text{ bytes.} \end{aligned}$$

Keep in mind that there are constant bytes that are not taken into account. In big-O notation, the computational complexity of the total memory consumption of RBAVO-DE can be represented as $O_{memory\_usage}(RBAVO - DE)$. This can then be calculated in big-O notation after eliminating all constants in the following way:

$$\begin{aligned} O_{memory\_usage}(RBAVO - DE) &= O_{memory\_usage}(\text{Input values memory usage}) + \\ &O_{memory\_usage}(\text{Additional memory usage}) = \\ O_{memory\_usage}(1) &+ (O_{memory\_usage}(N \times D) + O_{memory\_usage}(1) + O_{memory\_usage}(D)) = \\ &O_{memory\_usage}(N \times D). \end{aligned}$$

It can be difficult to compile a thorough comparison of the memory usage and execution time complexity of various meta-heuristic algorithms since these complexities might differ based on the particular implementation, size of the problem, and other operators. Furthermore, not all of the algorithms listed have comprehensive evaluations of the execution time and memory usage complexity available, and their properties may vary depending on the problem at hand.

5. Benefits and Drawbacks of the RBAVO-DE Algorithm

This part presents a balanced discussion of the benefits and drawbacks of the RBAVO-DE algorithm. The benefits of the RBAVO-DE algorithm can be listed as follows:

- **High Accuracy in Classification:** The RBAVO-DE algorithm has demonstrated high classification accuracy, achieving up to 100% in some cases. This is a noteworthy achievement, particularly regarding cancer datasets, as accurate gene selection can directly impact diagnostic and therapeutic outcomes.
- **Effective Feature Size Reduction:** The algorithm has shown remarkable capability in reducing the feature size by up to 98% while maintaining or improving classification accuracy. This dimensionality reduction is crucial for processing high-dimensional datasets efficiently.
- **Robustness Across Diverse Datasets:** The effectiveness of the RBAVO-DE algorithm across twenty-two cancer datasets indicates its robustness and adaptability to various genetic data characteristics. This versatility is beneficial for broader applications in genomics research.
- **Superior Performance Over Competitors:** When compared with binary variants of widely recognized meta-heuristic algorithms, RBAVO-DE stands out for its outstanding performance in accuracy and feature reduction, highlighting its innovative approach to gene selection.

On the other hand, the drawbacks of the proposed methodology can be listed as follows:

- **Computational Complexity:** The high dimensionality of the datasets and the iterative nature of meta-heuristic algorithms suggest that RBAVO-DE may have a significant computational cost. This aspect could limit its applicability in environments with constrained computational resources.
- **Generalizability Concerns:** Despite its success across various cancer datasets, the algorithm's generalizability to other types of biological data or diseases remains to be thoroughly investigated. It is crucial to test RBAVO-DE in broader contexts to confirm its applicability beyond the datasets examined.
- **Parameter Sensitivity:** Like many meta-heuristic algorithms, RBAVO-DE might be sensitive to its parameter settings, affecting performance and efficiency. Detailed studies on the impact of different parameter configurations and strategies for their optimization could enhance the algorithm's utility.

In summary, the RBAVO-DE algorithm represents a significant advancement in gene selection for cancer classification, with notable strengths in accuracy and efficiency. However, addressing its potential drawbacks through further research could broaden its applicability and improve its performance, making it an even more valuable tool in genomics and personalized medicine.

6. Conclusions and Future Directions

The RBAVO-DE approach proposed in this paper was the first to be implemented for handling GS issues in RNA-Seq gene expression data and determining potential biomarkers for different tumor classes to enhance the best solution discovered. The outcomes were promising, revealing that the proposed RBAVO-DE algorithm's effectiveness and capabilities were significantly improved. SVM and k -NN, two widely used classification models, were employed to evaluate the efficacy of each set of selected genes. The proposed RBAVO-DE algorithm's capability was compared to binary variants of eleven widely used

meta-heuristic techniques to assess it on different tumor classes with various instances. The assessment was executed employing a combination of evaluation measures, such as average fitness, classification accuracy, the number of selected genes, precision, recall, and F1-score. The proposed RBAVO-DE algorithm using the SVM and k -NN classifiers achieved more promising results than other optimizers in handling GS issues. Despite the promising outcomes, this research opens several avenues for future exploration to further advance the field:

- **Algorithmic Enhancements:** Improving the algorithm's efficiency and reducing computational complexity.
- **Cross-disease Applicability:** Testing the proposed algorithm on genetic data from various diseases beyond cancer.
- **Comparative Analyses with Deep Learning Models:** Evaluating RBAVO-DE against advanced deep learning models for genetic data analysis.
- **Real-world Clinical Validation:** Collaborating with clinical experts to validate the practical utility of selected genes in cancer treatment.
- **Scalability and Parallelization:** Enhancing the algorithm's scalability through parallel computing to handle larger genetic datasets efficiently.
- **Interdisciplinary Applications:** Exploring the algorithm's potential in other fields dealing with high-dimensional data, such as finance and environmental modeling.

In conclusion, while the RBAVO-DE algorithm represents a significant step forward in the field of gene selection for cancer classification, the paths outlined for future research highlight the potential for further advancements and broader applications of this work. Continued interdisciplinary collaboration and innovation will be crucial for unlocking the full capacity of gene selection methodologies to enhance healthcare outcomes and advance our understanding of complex diseases.

Author Contributions: M.G.G.: Resources, Validation, and Writing—review and editing. A.A.A.: Resources, Formal analysis, Data Curation, Validation, Writing—original draft, and Writing—review and editing. A.E.E.: Investigation, Visualization, Data Curation, Validation, Writing—original draft, and Writing—review and editing. A.A.A.E.-M.: Conceptualization, Methodology, Software, Formal analysis, Investigation, Data Curation, Validation, Writing—original draft, and Writing—review and editing. All authors have read and agreed to the published version of the manuscript.

Funding: Prince Sattam bin Abdulaziz University funded this research work under project number PSAU/2023/01/25612.

Data Availability Statement: For enhancing the applicability and reproducibility of this research, the developed software and relevant Python code of this research are publicly available and obtainable in [61].

Acknowledgments: The authors express their gratitude to Prince Sattam bin Abdulaziz University for providing financial support for this research project.

Conflicts of Interest: The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Abbreviations

The following abbreviations are used in this research:

AVO	African Vultures Optimization
HNSC	Head and Neck Squamous Cell Carcinoma
DE	Differential Evolution
KICH	Kidney Chromophobe
RBAVO-DE	Relief Binary African Vultures Optimization based on Differential Evolution
KIRP	Kidney Renal Papillary
GS	Gene Selection
LIHC	Liver Hepatocellular Carcinoma
RNA-Seq	RNA Sequencing

PAAD	Pancreatic Adenocarcinoma
DNA	Deoxyribonucleic Acid
PCPG	Pheochromocytoma and Paraganglioma
SVM	Support Vector Machine
READ	Rectum adenocarcinoma
<i>k</i> -NN	<i>k</i> -Nearest Neighbor
SARC	Sarcoma
FS	Feature Selection
SKCM	Skin Cutaneous Melanoma
MHMs	Meta-heuristic methods
$Mean_{Fi}$	Mean fitness
ML	Machine Learning
$Mean_{Ge}$	Mean size of chosen genes
DL	Deep learning
<i>STDE</i>	Standard Deviation
SA	Simulated Annealing
RF	Random Forest
FOIL	First-Order Inductive Learner
GSO	Gravitational Optimizer
SCCSA	Sine-Cosine-Cuckoo Search Algorithm
SMO	Spider Monkey Optimization
BHGSO	Binary Henry Gas Solubility Optimization
BPSO	Binary Particle Swarm Optimization
BABC	Binary Artificial Bee Colony
BSSA	Binary Salp Swarm Algorithm
BBA	Binary Bat Algorithm
BGWO	Binary Grey-Wolf Optimization
BGOA	Binary Grasshopper Optimization Algorithm
BWOA	Binary Whale Optimization Algorithm
BASO	Binary Atom Search Optimization
BBSA	Binary Bird Swarm Algorithm
BHHO	Binary Harris Hawks Optimization
BBBO	Binary Brown-Bear Optimization
BAO	Binary Aquila Optimization
BMOA	Binary Meerkat Optimization Algorithm
LUAD	Lung Adenocarcinoma
LUSC	Lung Squamous Cell Carcinoma
BRCA	Breast Invasive Carcinoma
KIRC	Kidney Renal Clear Cell Carcinoma
UCEC	Uterine Corpus Endometrial Carcinoma
RPKM	Reads Per Kilobase per Million mapped reads
STAD	Stomach Adenocarcinoma
SDAE	Stacked Denoising Autoencoder
CGA	Cancer Genome Atlas
BLCA	Bladder Urothelial Carcinoma
THCA	Thyroid Cancer
CESC	Cervical and Endocervical Cancers
CHOL	Cholangiocarcinoma
COAD	Colon Adenocarcinoma
ESCA	Esophageal Cancer
GBM	Glioblastoma Multiforme
THYM	Thymoma

Appendix A. Comparison Results of the Proposed RBAVO-DE and Its Counterparts

Appendix A.1. Comparison Results Based on the k -NN Model

Table A1. The results of the proposed RBAVO-DE algorithm and its counterparts employing k -NN regarding average fitness values.

Dataset	Metric	RBAVO-DE	BSSA	BABC	BPSO	BBA	BGWO	BWOA	BGOA	BHHO	BBSA	BASO	BHGSO
BLCA	Average	0.00190	0.01110	0.00710	0.01120	0.01470	0.00940	0.01070	0.01140	0.01310	0.01030	0.01080	0.01590
	STDE	0.00430	0.00530	0.00440	0.00540	0.00240	0.00550	0.00540	0.00540	0.00440	0.00550	0.00540	0.00180
CESC	Average	0.01610	0.01990	0.02020	0.02050	0.01980	0.02000	0.02020	0.02020	0.01970	0.02020	0.02030	0.02060
	STDE	0.00000	0.00020	0.00010	0.00010	0.00020	0.00020	0.00020	0.00010	0.00010	0.00010	0.00010	0.00010
CHOL	Average	0.00000	0.00390	0.00430	0.00450	0.00390	0.00400	0.00430	0.00430	0.00370	0.00420	0.00420	0.00470
	STDE	0.00000	0.00020	0.00010	0.00010	0.00020	0.00020	0.00020	0.00010	0.00010	0.00010	0.00010	0.00010
COAD	Average	0.00000	0.00390	0.00430	0.00450	0.00390	0.00400	0.00430	0.00430	0.00370	0.00420	0.00420	0.00470
	STDE	0.00000	0.00020	0.00010	0.00010	0.00020	0.00020	0.00020	0.00010	0.00010	0.00010	0.00010	0.00010
ESCA	Average	0.02560	0.02930	0.02960	0.02990	0.02930	0.02940	0.02970	0.02960	0.02910	0.02960	0.02980	0.03000
	STDE	0.00000	0.00020	0.00010	0.00010	0.00020	0.00020	0.00020	0.00010	0.00010	0.00010	0.00010	0.00020
GBM	Average	0.00000	0.00390	0.00430	0.00450	0.00390	0.00400	0.00430	0.00430	0.00370	0.00420	0.00420	0.00470
	STDE	0.00000	0.00020	0.00010	0.00010	0.00020	0.00020	0.00020	0.00010	0.00010	0.00010	0.00010	0.00010
HNSC	Average	0.00710	0.01250	0.01290	0.01330	0.01280	0.01240	0.01310	0.01320	0.01270	0.01290	0.01320	0.01360
	STDE	0.00350	0.00160	0.00140	0.00010	0.00030	0.00210	0.00010	0.00010	0.00020	0.00150	0.00010	0.00010
KICH	Average	0.00000	0.00390	0.00430	0.00450	0.00390	0.00400	0.00430	0.00430	0.00370	0.00420	0.00420	0.00470
	STDE	0.00000	0.00020	0.00010	0.00010	0.00020	0.00020	0.00020	0.00010	0.00010	0.00010	0.00010	0.00010
KIRC	Average	0.00250	0.01110	0.01040	0.01220	0.01200	0.01020	0.01200	0.01070	0.01040	0.01120	0.01170	0.01290
	STDE	0.00380	0.00270	0.00340	0.00190	0.00020	0.00340	0.00190	0.00320	0.00300	0.00280	0.00220	0.00010
KIRP	Average	0.00000	0.00390	0.00430	0.00450	0.00390	0.00400	0.00430	0.00430	0.00370	0.00420	0.00420	0.00470
	STDE	0.00000	0.00020	0.00010	0.00010	0.00020	0.00020	0.00020	0.00010	0.00010	0.00010	0.00010	0.00010
LIHC	Average	0.00000	0.00390	0.00430	0.00450	0.00390	0.00400	0.00420	0.00430	0.00380	0.00430	0.00420	0.00460
	STDE	0.00000	0.00020	0.00010	0.00010	0.00020	0.00020	0.00020	0.00010	0.00010	0.00010	0.00010	0.00010
LUAD	Average	0.00870	0.01260	0.01290	0.01310	0.01240	0.01270	0.01290	0.01290	0.01240	0.01290	0.01300	0.01340
	STDE	0.00000	0.00020	0.00010	0.00010	0.00020	0.00020	0.00010	0.00010	0.00020	0.00010	0.00010	0.00010
LUSC	Average	0.00000	0.00400	0.00430	0.00450	0.00380	0.00410	0.00430	0.00430	0.00390	0.00430	0.00430	0.00480
	STDE	0.00000	0.00020	0.00010	0.00010	0.00020	0.00010	0.00020	0.00010	0.00010	0.00010	0.00010	0.00010
PAAD	Average	0.00000	0.00390	0.00430	0.00450	0.00390	0.00400	0.00430	0.00430	0.00370	0.00420	0.00420	0.00470
	STDE	0.00000	0.00020	0.00010	0.00010	0.00020	0.00020	0.00020	0.00010	0.00010	0.00010	0.00010	0.00010
PCPG	Average	0.00000	0.00390	0.00430	0.00450	0.00390	0.00400	0.00430	0.00430	0.00370	0.00420	0.00420	0.00470
	STDE	0.00000	0.00020	0.00010	0.00010	0.00020	0.00020	0.00020	0.00010	0.00010	0.00010	0.00010	0.00010
READ	Average	0.00000	0.00390	0.00430	0.00450	0.00390	0.00400	0.00430	0.00430	0.00370	0.00420	0.00420	0.00470
	STDE	0.00000	0.00020	0.00010	0.00010	0.00020	0.00020	0.00020	0.00010	0.00010	0.00010	0.00010	0.00010
SARC	Average	0.01890	0.02260	0.02290	0.02320	0.02260	0.02270	0.02300	0.02290	0.02240	0.02290	0.02300	0.02330
	STDE	0.00000	0.00020	0.00010	0.00010	0.00020	0.00020	0.00020	0.00010	0.00010	0.00010	0.00010	0.00010
SKCM	Average	0.00000	0.00390	0.00430	0.00450	0.00390	0.00400	0.00430	0.00430	0.00370	0.00420	0.00420	0.00470
	STDE	0.00000	0.00020	0.00010	0.00010	0.00020	0.00020	0.00020	0.00010	0.00010	0.00010	0.00010	0.00010
STAD	Average	0.00070	0.00830	0.00620	0.00930	0.01470	0.00810	0.00950	0.00800	0.01170	0.00940	0.00840	0.01570
	STDE	0.00280	0.00500	0.00360	0.00510	0.00020	0.00500	0.00510	0.00480	0.00480	0.00520	0.00490	0.00010
THCA	Average	0.00000	0.00390	0.00430	0.00450	0.00390	0.00400	0.00430	0.00430	0.00370	0.00420	0.00420	0.00460
	STDE	0.00000	0.00020	0.00010	0.00010	0.00020	0.00020	0.00020	0.00010	0.00010	0.00010	0.00010	0.00010
THYM	Average	0.00000	0.00390	0.00430	0.00450	0.00390	0.00400	0.00430	0.00430	0.00370	0.00420	0.00420	0.00470
	STDE	0.00000	0.00020	0.00010	0.00010	0.00020	0.00020	0.00020	0.00010	0.00010	0.00010	0.00010	0.00010
UCEC	Average	0.00000	0.01920	0.01710	0.02780	0.03540	0.02010	0.02770	0.02690	0.02560	0.02520	0.02860	0.03670
	STDE	0.00250	0.01200	0.01220	0.00610	0.01000	0.01190	0.00620	0.00720	0.00830	0.00910	0.00450	0.01050
Ranking	W T L	22 0 0	0 0 22	0 0 22	0 0 22	0 0 22	0 0 22	0 0 22	0 0 22	0 0 22	0 0 22	0 0 22	0 0 22

Boldface values denote the best results.

Table A2. The results of the proposed RBAVO-DE algorithm and its counterparts employing k -NN regarding average values of selected genes.

Dataset	Metric	RBAVO-DE	BSSA	BABC	BPSO	BBA	BGWO	BWOA	BGOA	BHHO	BBSA	BASO	BHGSO
BLCA	Average	142.03	210.17	237.70	235.00	196.03	218.17	228.50	226.30	195.23	228.27	232.03	237.13
	STDE	017.18	020.50	019.07	017.12	023.83	018.27	019.43	014.81	013.17	014.60	018.90	014.14
CESC	Average	122.50	195.13	213.03	225.30	194.10	199.50	213.63	212.67	185.60	213.33	215.90	233.43
	STDE	005.45	011.17	005.31	003.11	009.66	009.30	008.35	005.66	006.49	005.88	006.02	005.64
CHOL	Average	122.13	195.13	213.03	225.30	194.10	199.50	213.63	212.67	185.60	211.00	211.57	232.63
	STDE	006.18	011.17	005.31	003.11	009.66	009.30	008.35	005.66	006.49	006.44	005.08	004.36
COAD	Average	120.30	195.13	213.03	225.30	194.10	199.50	213.63	212.67	185.60	211.00	211.57	232.63
	STDE	005.82	011.17	005.31	003.11	009.66	009.30	008.35	005.66	006.49	006.44	005.08	004.36
ESCA	Average	120.17	195.13	213.03	225.30	194.10	199.50	213.63	212.67	185.60	211.57	218.87	230.07
	STDE	006.81	011.17	005.31	003.11	009.66	009.30	008.35	005.66	006.49	006.94	004.68	007.64
GBM	Average	120.73	195.13	212.77	225.30	194.10	199.73	213.17	212.80	185.43	211.00	211.83	232.63
	STDE	006.54	011.17	005.40	003.11	009.66	009.05	008.51	005.72	006.40	006.44	004.99	004.39
HNSC	Average	133.43	202.87	222.67	227.43	203.00	211.37	217.97	220.87	195.93	219.83	223.63	241.27
	STDE	017.63	009.26	011.96	003.91	013.24	007.65	007.41	005.32	008.26	007.71	005.68	006.21
KICH	Average	121.43	195.13	213.03	225.30	194.10	199.50	213.63	212.67	185.60	211.00	211.57	232.63
	STDE	006.58	011.17	005.31	003.11	009.66	009.30	008.35	005.66	006.49	006.44	005.08	004.36

Table A2. *Cont.*

Dataset	Metric	RBAVO-DE	BSSA	BABC	BPSO	BBA	BGWO	BWOA	BGOA	BHHO	BBSA	BASO	BHGSO
KIRC	Average	137.57	200.83	220.17	227.70	192.17	209.87	216.13	222.30	193.27	218.73	218.60	234.10
	STDE	017.10	009.63	011.74	008.27	008.73	014.08	010.11	015.18	017.36	013.82	012.78	004.21
KIRP	Average	122.33	195.13	213.03	225.30	194.10	199.50	213.63	212.67	185.60	211.00	211.57	232.63
	STDE	009.08	011.17	005.31	003.11	009.66	009.30	008.35	005.66	006.49	006.44	005.08	004.36
LIHC	Average	123.40	197.00	214.03	225.67	193.33	200.63	212.33	213.63	189.30	213.10	212.03	232.43
	STDE	006.59	007.78	005.22	003.36	009.17	010.02	009.35	004.89	006.89	005.69	004.81	004.97
LUAD	Average	127.03	197.47	216.13	222.93	190.83	205.33	212.57	215.67	191.80	215.27	217.83	237.90
	STDE	008.62	008.98	004.74	006.16	008.61	010.57	006.15	004.43	008.50	006.13	004.71	004.94
LUSC	Average	125.73	198.87	215.63	223.00	189.13	203.60	213.23	216.37	194.67	215.53	215.23	241.23
	STDE	006.35	008.77	003.54	004.89	010.69	007.20	009.70	003.44	006.40	005.54	006.16	004.66
PAAD	Average	121.10	195.13	213.03	225.30	194.10	199.50	213.63	212.67	185.60	211.00	211.57	232.63
	STDE	007.71	011.17	005.31	003.11	009.66	009.30	008.35	005.66	006.49	006.44	005.08	004.36
PCPG	Average	122.17	195.13	213.03	225.30	194.10	199.50	213.63	212.67	185.60	211.00	211.57	232.63
	STDE	006.42	011.17	005.31	003.11	009.66	009.30	008.35	005.66	006.49	006.44	005.08	004.36
READ	Average	120.13	195.13	213.03	225.30	194.10	199.50	213.63	212.67	185.60	211.00	211.57	232.63
	STDE	007.72	011.17	005.31	003.11	009.66	009.30	008.35	005.66	006.49	006.44	005.08	004.36
SARC	Average	122.47	195.13	213.03	225.30	194.10	199.50	213.63	212.67	185.60	212.90	216.43	231.80
	STDE	005.79	011.17	005.31	003.11	009.66	009.30	008.35	005.66	006.49	006.30	004.98	003.40
SKCM	Average	123.03	195.13	213.03	225.30	194.10	199.50	213.63	212.67	185.60	211.00	211.57	232.63
	STDE	008.83	011.17	005.31	003.11	009.66	009.30	008.35	005.66	006.49	006.44	005.08	004.36
STAD	Average	142.60	215.00	236.17	243.33	187.50	221.17	234.87	232.77	201.73	231.13	237.10	236.97
	STDE	013.31	018.43	013.94	016.29	010.27	019.24	022.63	014.25	014.00	014.04	015.35	005.86
THCA	Average	117.97	195.43	213.40	225.13	194.10	199.63	214.13	212.80	186.53	211.00	211.57	231.90
	STDE	006.67	011.20	005.29	003.10	009.66	009.81	008.59	005.69	005.88	006.44	005.08	004.78
THYM	Average	120.37	195.13	213.03	225.30	194.10	199.50	213.63	212.67	185.60	211.00	211.57	232.63
	STDE	004.89	011.17	005.31	003.11	009.66	009.30	008.35	005.66	006.49	006.44	005.08	004.36
UCEC	Average	143.93	216.33	237.87	236.00	242.37	220.07	230.80	229.17	206.77	228.40	235.00	266.67
	STDE	013.18	015.41	015.82	010.11	034.13	011.26	011.27	011.66	010.84	010.32	006.14	030.55
Ranking	W T L	22 0 0	0 0 22	0 0 22	0 0 22	0 0 22	0 0 22	0 0 22	0 0 22	0 0 22	0 0 22	0 0 22	0 0 22

Boldface values denote the best results.

Table A3. The results of the proposed RBAVO-DE algorithm and its counterparts employing k -NN regarding average precision values.

Dataset	RBAVO-DE	BSSA	BABC	BPSO	BBA	BGWO	BWOA	BGOA	BHHO	BBSA	BASO	BHGSO
BLCA	1.00000	1.00000	1.00000	1.00000	0.99960	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
CESC	0.98390	0.98390	0.98390	0.98390	0.98390	0.98390	0.98390	0.98390	0.98390	0.98390	0.98390	0.98390
CHOL	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
COAD	1.00000	1.00000	1.00000	1.00000	0.99890	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
ESCA	0.97370	0.97370	0.97370	0.97370	0.97370	0.97370	0.97370	0.97370	0.97370	0.97370	0.97370	0.97370
GBM	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
HNSC	0.99180	0.96470	0.96540	0.96500	0.96350	0.96690	0.96660	0.96570	0.96540	0.96540	0.96100	0.96290
KICH	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
KIRC	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
KIRP	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
LIHC	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
LUAD	1.00000	0.99770	0.99740	0.99840	0.99870	0.99770	0.99800	0.99540	0.99710	0.99740	0.99670	0.99800
LUSC	1.00000	1.00000	1.00000	1.00000	0.99790	1.00000	0.99970	1.00000	1.00000	1.00000	0.97900	0.99930
PAAD	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
PCPG	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
READ	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
SARC	0.98110	0.98110	0.98110	0.98110	0.98110	0.98110	0.98110	0.98110	0.98110	0.98110	0.98110	0.98110
SKCM	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
STAD	0.99920	0.95180	0.95180	0.95180	0.94870	0.95180	0.95180	0.95180	0.95180	0.95180	0.99000	0.95090
THCA	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
THYM	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
UCEC	1.00000	1.00000	1.00000	1.00000	0.99910	1.00000	1.00000	1.00000	1.00000	1.00000	0.93330	0.99910
Ranking (W T L)	3 19 0	0 19 3	0 19 3	0 19 3	0 15 7	0 19 3	0 18 4	0 19 3	0 19 3	0 19 3	0 17 5	0 17 5

Boldface values denote the best results.

Table A4. The results of the proposed RBAVO-DE algorithm and its counterparts employing k -NN regarding average recall values.

[illegible]

Table A4. Cont.

Dataset	RBAVO-DE	BSSA	BABC	BPSO	BBA	BGWO	BWOA	BGOA	BHHO	BBSA	BASO	BHGSO
HNSC	1.00000	0.99520	0.99450	0.99480	0.99590	0.99280	0.99310	0.99420	0.99450	0.99450	0.99790	0.99730
KICH	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
KIRC	0.99710	0.99040	0.99040	0.99040	0.98970	0.99040	0.99040	0.99040	0.99040	0.99040	0.99010	0.99040
KIRP	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
LIHC	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
LUAD	1.00000	0.99240	0.99270	0.99170	0.99140	0.99240	0.99210	0.99470	0.99310	0.99270	0.99340	0.99210
LUSC	1.00000	1.00000	1.00000	1.00000	0.99580	1.00000	1.00000	1.00000	0.99930	1.00000	0.99310	0.99270
PAAD	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
PCPG	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
READ	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
SARC	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
SKCM	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
STAD	1.00000	1.00000	1.00000	1.00000	0.99750	1.00000	1.00000	1.00000	1.00000	1.00000	0.99620	0.99750
THCA	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
THYM	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
UCEC	1.00000	1.00000	1.00000	1.00000	0.99370	1.00000	1.00000	1.00000	1.00000	1.00000	0.99640	1.00000
Ranking (W T L)	3 19 0	0 19 3	0 19 3	0 19 3	0 16 6	0 19 3	0 19 3	0 19 3	0 18 4	0 19 3	0 16 6	0 17 5

Boldface values denote the best results.

Table A5. The results of the proposed RBAVO-DE algorithm and its counterparts employing k -NN regarding average F1-score values.

Dataset	RBAVO-DE	BSSA	BABC	BPSO	BBA	BGWO	BWOA	BGOA	BHHO	BBSA	BASO	BHGSO
BLCA	1.00000	1.00000	1.00000	1.00000	0.99980	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
CESC	0.99190	0.99190	0.99190	0.99190	0.99190	0.99190	0.99190	0.99190	0.99190	0.99190	0.99190	0.99190
CHOL	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
COAD	1.00000	1.00000	1.00000	1.00000	0.99940	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
ESCA	0.98670	0.98670	0.98670	0.98670	0.98670	0.98670	0.98670	0.98670	0.98670	0.98670	0.98670	0.98670
GBM	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
HNSC	0.99590	0.97970	0.97970	0.97970	0.97940	0.97970	0.97970	0.97970	0.97970	0.97970	0.97980	0.97970
KICH	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
KIRC	0.99860	0.99520	0.99520	0.99520	0.99480	0.99520	0.99520	0.99520	0.99520	0.99520	0.99500	0.99520
KIRP	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
LIHC	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
LUAD	0.99500	0.99500	0.99500	0.99500	0.99500	0.99500	0.99500	0.99500	0.99500	0.99500	0.99500	0.99500
LUSC	1.00000	1.00000	1.00000	1.00000	0.99690	1.00000	0.99980	1.00000	0.99970	1.00000	0.99630	0.99600
PAAD	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
PCPG	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
READ	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
SARC	0.99050	0.99050	0.99050	0.99050	0.99050	0.99050	0.99050	0.99050	0.99050	0.99050	0.99050	0.99050
SKCM	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
STAD	0.99960	0.97530	0.97530	0.97530	0.97240	0.97530	0.97530	0.97530	0.97530	0.97530	0.97180	0.97360
THCA	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
THYM	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
UCEC	1.00000	1.00000	1.00000	1.00000	0.99640	1.00000	1.00000	1.00000	1.00000	1.00000	0.99640	0.99960
Ranking (W T L)	3 19 0	0 18 4	0 18 4	0 18 4	0 14 8	0 18 4	0 17 5	0 18 4	0 17 5	0 18 4	0 16 6	0 16 6

Boldface values denote the best results.

Appendix A.2. Comparison Results Based on the SVM Model

Table A6. The results of the proposed RBAVO-DE algorithm and its counterparts employing SVM regarding average fitness values.

Dataset	Metric	RBAVO-DE	BSSA	BABC	BPSO	BBA	BGWO	BWOA	BGOA	BHHO	BBSA	BASO	BHGSO
BLCA	Average	0.00000	0.00390	0.00430	0.00450	0.00390	0.00400	0.00430	0.00430	0.00370	0.00420	0.00420	0.00460
	STDE	0.00000	0.00020	0.00010	0.00010	0.00020	0.00020	0.00020	0.00010	0.00010	0.00010	0.00010	0.00010
CESC	Average	0.00000	0.00390	0.00430	0.00450	0.00390	0.00400	0.00430	0.00430	0.00370	0.00420	0.00420	0.00470
	STDE	0.00000	0.00020	0.00010	0.00010	0.00020	0.00020	0.00020	0.00010	0.00010	0.00010	0.00010	0.00010
CHOL	Average	0.00000	0.00390	0.00430	0.00450	0.00390	0.00400	0.00430	0.00430	0.00370	0.00420	0.00420	0.00470
	STDE	0.00000	0.00020	0.00010	0.00010	0.00020	0.00020	0.00020	0.00010	0.00010	0.00010	0.00010	0.00010
COAD	Average	0.00000	0.00390	0.00430	0.00450	0.00390	0.00400	0.00430	0.00430	0.00370	0.00420	0.00420	0.00470
	STDE	0.00000	0.00020	0.00010	0.00010	0.00020	0.00020	0.00020	0.00010	0.00010	0.00010	0.00010	0.00010
ESCA	Average	0.01370	0.02840	0.02720	0.02990	0.02930	0.02690	0.02970	0.02880	0.02830	0.02880	0.02980	0.03000
	STDE	0.01280	0.00460	0.00750	0.00010	0.00020	0.00750	0.00020	0.00450	0.00460	0.00440	0.00010	0.00020
GBM	Average	0.00000	0.00390	0.00430	0.00450	0.00390	0.00400	0.00430	0.00430	0.00370	0.00420	0.00420	0.00460
	STDE	0.00000	0.00020	0.00010	0.00010	0.00020	0.00020	0.00020	0.00010	0.00010	0.00010	0.00010	0.00010
HNSC	Average	0.00000	0.00390	0.00430	0.00450	0.00390	0.00400	0.00430	0.00430	0.00370	0.00420	0.00420	0.00460
	STDE	0.00000	0.00020	0.00010	0.00010	0.00020	0.00020	0.00020	0.00010	0.00010	0.00010	0.00010	0.00010

Table A6. Cont.

Dataset	Metric	RBAVO-DE	BSSA	BABC	BPSO	BBA	BGWO	BWOA	BGOA	BHHO	BBSA	BASO	BHGSO
KICH	Average	0.00000	0.00390	0.00430	0.00450	0.00390	0.00400	0.00430	0.00430	0.00370	0.00420	0.00420	0.00470
	STDE	0.00000	0.00020	0.00010	0.00010	0.00020	0.00020	0.00020	0.00010	0.00010	0.00010	0.00010	0.00010
KIRC	Average	0.00000	0.00390	0.00430	0.00450	0.00380	0.00400	0.00430	0.00430	0.00380	0.00430	0.00430	0.00460
	STDE	0.00000	0.00020	0.00010	0.00010	0.00020	0.00020	0.00020	0.00010	0.00010	0.00010	0.00010	0.00020
KIRP	Average	0.00000	0.00390	0.00430	0.00450	0.00390	0.00400	0.00430	0.00430	0.00370	0.00420	0.00420	0.00470
	STDE	0.00000	0.00020	0.00010	0.00010	0.00020	0.00020	0.00020	0.00010	0.00010	0.00010	0.00010	0.00010
LIHC	Average	0.00000	0.00410	0.00440	0.00450	0.00500	0.00430	0.00440	0.00450	0.00400	0.00450	0.00450	0.00500
	STDE	0.00000	0.00010	0.00010	0.00010	0.00280	0.00010	0.00020	0.00010	0.00020	0.00010	0.00010	0.00020
LUAD	Average	0.00840	0.01250	0.01290	0.01310	0.01250	0.01260	0.01290	0.01290	0.01230	0.01290	0.01260	0.01330
	STDE	0.00160	0.00020	0.00010	0.00010	0.00020	0.00020	0.00020	0.00010	0.00010	0.00010	0.00150	0.00010
LUSC	Average	0.00000	0.00390	0.00430	0.00450	0.00390	0.00400	0.00430	0.00430	0.00370	0.00420	0.00420	0.00470
	STDE	0.00000	0.00020	0.00010	0.00010	0.00020	0.00020	0.00020	0.00010	0.00010	0.00010	0.00010	0.00010
PAAD	Average	0.00000	0.00390	0.00430	0.00450	0.00390	0.00400	0.00430	0.00430	0.00370	0.00420	0.00420	0.00470
	STDE	0.00000	0.00020	0.00010	0.00010	0.00020	0.00020	0.00020	0.00010	0.00010	0.00010	0.00010	0.00010
PCPG	Average	0.00000	0.00390	0.00430	0.00450	0.00390	0.00400	0.00430	0.00430	0.00370	0.00420	0.00420	0.00470
	STDE	0.00000	0.00020	0.00010	0.00010	0.00020	0.00020	0.00020	0.00010	0.00010	0.00010	0.00010	0.00010
READ	Average	0.00000	0.00390	0.00430	0.00450	0.00390	0.00400	0.00430	0.00430	0.00370	0.00420	0.00420	0.00470
	STDE	0.00000	0.00020	0.00010	0.00010	0.00020	0.00020	0.00020	0.00010	0.00010	0.00010	0.00010	0.00010
SARC	Average	0.00570	0.01780	0.01570	0.01840	0.02260	0.01480	0.01760	0.01870	0.02000	0.01870	0.02000	0.02330
	STDE	0.00860	0.00800	0.00880	0.00800	0.00020	0.00900	0.00820	0.00770	0.00610	0.00770	0.00670	0.00010
SKCM	Average	0.00000	0.00390	0.00430	0.00450	0.00390	0.00400	0.00430	0.00430	0.00370	0.00420	0.00420	0.00470
	STDE	0.00000	0.00020	0.00010	0.00010	0.00020	0.00020	0.00020	0.00010	0.00010	0.00010	0.00010	0.00010
STAD	Average	0.01040	0.01490	0.01530	0.01550	0.01450	0.01500	0.01530	0.01530	0.01470	0.01520	0.01530	0.01560
	STDE	0.00280	0.00020	0.00010	0.00010	0.00020	0.00020	0.00020	0.00010	0.00010	0.00010	0.00010	0.00010
THCA	Average	0.00000	0.00390	0.00430	0.00450	0.00390	0.00400	0.00430	0.00430	0.00370	0.00420	0.00420	0.00470
	STDE	0.00000	0.00020	0.00010	0.00010	0.00020	0.00020	0.00020	0.00010	0.00010	0.00010	0.00010	0.00010
THYM	Average	0.00000	0.00390	0.00430	0.00450	0.00390	0.00400	0.00430	0.00430	0.00370	0.00420	0.00420	0.00470
	STDE	0.00000	0.00020	0.00010	0.00010	0.00020	0.00020	0.00020	0.00010	0.00010	0.00010	0.00010	0.00010
UCEC	Average	0.00000	0.00390	0.00430	0.00450	0.00380	0.00400	0.00430	0.00430	0.00380	0.00420	0.00430	0.00470
	STDE	0.00000	0.00020	0.00010	0.00010	0.00020	0.00020	0.00020	0.00010	0.00020	0.00010	0.00010	0.00010
Ranking	W T L	22 0 0	0 0 22	0 0 22	0 0 22	0 0 22	0 0 22	0 0 22	0 0 22	0 0 22	0 0 22	0 0 22	0 0 22

Boldface values denote the best results.

Table A7. The results of the proposed RBAVO-DE algorithm and its counterparts employing SVM regarding average chosen gene values.

Dataset	Metric	RBAVO-DE	BSSA	BABC	BPSO	BBA	BGWO	BWOA	BGOA	BHHO	BBSA	BASO	BHGSO
BLCA	Average	121.37	194.70	213.03	225.30	194.10	199.50	213.40	212.67	186.27	211.00	211.83	232.37
	STDE	007.49	011.44	005.18	003.11	009.66	009.30	008.44	005.66	005.77	006.44	005.25	004.42
CESC	Average	119.83	195.13	213.03	225.30	194.10	199.50	213.63	212.67	185.60	211.00	211.57	232.63
	STDE	007.02	011.17	005.31	003.11	009.66	009.30	008.35	005.66	006.49	006.44	005.08	004.36
CHOL	Average	125.57	195.13	213.03	225.30	194.10	199.50	213.63	212.67	185.60	211.00	211.57	232.63
	STDE	007.14	011.17	005.31	003.11	009.66	009.30	008.35	005.66	006.49	006.44	005.08	004.36
COAD	Average	122.63	195.13	213.03	225.30	194.10	199.50	213.63	212.67	185.60	211.00	211.57	232.63
	STDE	007.73	011.17	005.31	003.11	009.66	009.30	008.35	005.66	006.49	006.44	005.08	004.36
ESCA	Average	125.50	195.13	215.40	225.30	194.10	201.77	213.63	213.13	185.70	212.40	218.87	230.07
	STDE	011.18	011.17	009.58	003.11	009.66	012.33	008.35	006.44	006.47	008.98	004.68	007.64
GBM	Average	119.73	194.93	213.03	225.30	193.90	198.77	213.70	212.83	185.50	211.00	211.00	232.33
	STDE	008.07	011.27	005.31	003.11	009.47	010.28	008.36	005.69	006.51	006.44	004.92	004.15
HNSC	Average	122.17	196.40	213.90	225.13	194.03	202.10	213.83	213.50	186.93	211.47	210.87	232.17
	STDE	008.29	008.49	005.22	003.19	009.56	010.76	009.34	005.22	005.74	005.66	004.79	005.02
KICH	Average	121.73	195.13	213.03	225.30	194.10	199.50	213.63	212.67	185.60	211.00	211.57	232.63
	STDE	008.49	011.17	005.31	003.11	009.66	009.30	008.35	005.66	006.49	006.44	005.08	004.36
KIRC	Average	123.87	197.10	214.70	225.00	191.93	200.30	212.97	213.13	187.57	213.83	214.63	231.27
	STDE	007.57	009.42	005.09	004.23	008.70	011.59	007.64	004.94	005.31	005.81	004.20	008.02
KIRP	Average	120.07	195.13	213.03	225.30	194.10	199.50	213.63	212.67	185.60	211.00	211.57	232.63
	STDE	007.87	011.17	005.31	003.11	009.66	009.30	008.35	005.66	006.49	006.44	005.08	004.36
LIHC	Average	122.63	204.27	220.37	226.40	212.83	214.07	219.07	223.70	200.60	223.03	225.17	249.90
	STDE	008.01	006.99	006.29	006.80	017.60	007.06	007.88	004.13	007.66	006.48	004.81	007.97
LUAD	Average	121.17	195.13	213.03	225.30	194.10	199.43	213.63	212.67	185.87	212.40	214.50	233.00
	STDE	007.56	011.17	005.31	003.11	009.66	009.37	008.35	005.66	006.40	006.10	005.98	004.34
LUSC	Average	122.67	195.13	213.03	225.30	194.10	199.50	213.63	212.67	185.60	211.00	211.57	232.63
	STDE	006.28	011.17	005.31	003.11	009.66	009.30	008.35	005.66	006.49	006.44	005.08	004.36
PAAD	Average	119.23	195.13	213.03	225.30	194.10	199.50	213.63	212.67	185.60	211.00	211.57	232.63
	STDE	006.90	011.17	005.31	003.11	009.66	009.30	008.35	005.66	006.49	006.44	005.08	004.36
PCPG	Average	122.43	195.13	213.03	225.30	194.10	199.50	213.63	212.67	185.60	211.00	211.57	232.63
	STDE	006.68	011.17	005.31	003.11	009.66	009.30	008.35	005.66	006.49	006.44	005.08	004.36
READ	Average	123.93	195.13	213.03	225.30	194.10	199.50	213.63	212.67	185.60	211.00	211.57	232.63
	STDE	007.45	011.17	005.31	003.11	009.66	009.30	008.35	005.66	006.49	006.44	005.08	004.36
SARC	Average	135.27	202.90	226.90	233.77	194.10	210.53	223.87	217.80	189.93	220.57	220.87	231.53
	STDE	018.57	020.86	020.25	014.44	009.66	017.74	017.39	013.90	013.41	016.40	011.76	003.70
SKCM	Average	122.47	195.13	213.03	225.30	194.10	199.50	213.63	212.67	185.60	211.00	211.57	232.63
	STDE	007.66	011.17	005.31	003.11	009.66	009.30	008.35	005.66	006.49	006.44	005.08	004.36

Table A7. Cont.

Dataset	Metric	RBAVO-DE	BSSA	BABC	BPSO	BBA	BGWO	BWOA	BGOA	BHHO	BBSA	BASO	BHGSO
STAD	Average	123.53	194.60	213.40	225.03	194.30	198.80	214.10	212.70	185.70	211.97	215.07	231.77
	STDE	010.65	010.22	005.38	003.29	009.79	009.09	008.68	005.54	006.92	005.88	005.12	004.58
THCA	Average	119.47	195.13	213.03	225.30	194.10	199.50	213.63	212.67	185.60	211.00	211.57	232.63
	STDE	005.58	011.17	005.31	003.11	009.66	009.30	008.35	005.66	006.49	006.44	005.08	004.36
THYM	Average	121.80	195.13	213.03	225.30	194.10	199.50	213.63	212.67	185.60	211.00	211.57	232.63
	STDE	007.21	011.17	005.31	003.11	009.66	009.30	008.35	005.66	006.49	006.44	005.08	004.36
UCEC	Average	122.23	196.00	213.27	224.83	192.03	200.97	212.60	213.03	189.23	212.23	212.93	233.93
	STDE	005.93	009.06	006.31	004.27	007.85	008.24	007.74	005.67	008.78	004.98	004.97	003.97
Ranking	W T L	22 0 0	0 0 22	0 0 22	0 0 22	0 0 22	0 0 22	0 0 22	0 0 22	0 0 22	0 0 22	0 0 22	0 0 22

Boldface values denote the best results.

Table A8. The results of the proposed RBAVO-DE algorithm and its counterparts employing SVM regarding average precision values.

Dataset	RBAVO-DE	BSSA	BABC	BPSO	BBA	BGWO	BWOA	BGOA	BHHO	BBSA	BASO	BHGSO
BLCA	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
CESC	1.00000	1.00000	1.00000	1.00000	1.00000	0.99890	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
CHOL	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
COAD	1.00000	1.00000	1.00000	1.00000	0.99890	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
ESCA	1.00000	1.00000	1.00000	1.00000	0.99820	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	0.99210
GBM	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
HNSC	1.00000	0.94140	0.94170	0.93750	0.92840	0.94140	0.93990	0.94170	0.94110	0.94050	0.84550	0.93150
KICH	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
KIRC	1.00000	1.00000	1.00000	1.00000	0.99940	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
KIRP	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
LIHC	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
LUAD	1.00000	1.00000	1.00000	1.00000	0.99870	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	0.99930
LUSC	1.00000	0.99860	0.99860	0.99860	0.99830	0.99660	0.99860	0.99900	0.99930	0.99860	0.81400	0.99970
PAAD	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
PCPG	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
READ	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
SARC	1.00000	0.99940	1.00000	0.99940	0.99500	0.99940	1.00000	0.99940	0.99620	0.99940	1.00000	0.99120
SKCM	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
STAD	0.98830	0.94840	0.95110	0.94170	0.93610	0.94730	0.94690	0.94690	0.93910	0.94650	0.95200	0.93500
THCA	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
THYM	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
UCEC	1.00000	0.99910	0.99820	0.99820	0.99300	1.00000	0.99820	0.99820	0.99910	1.00000	1.00000	0.99390
Ranking (W T L)	3 19 0	0 17 5	0 18 4	0 17 5	0 12 10	0 18 4	0 18 4	0 17 5	0 17 5	0 18 4	0 19 3	0 15 7

Boldface values denote the best results.

Table A9. The results of the proposed RBAVO-DE algorithm and its counterparts employing SVM regarding average recall values.

Dataset	RBAVO-DE	BSSA	BABC	BPSO	BBA	BGWO	BWOA	BGOA	BHHO	BBSA	BASO	BHGSO
BLCA	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
CESC	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
CHOL	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
COAD	1.00000	1.00000	1.00000	1.00000	0.99940	1.00000	1.00000	1.00000	1.00000	1.00000	0.99940	1.00000
ESCA	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
GBM	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
HNSC	1.00000	1.00000	1.00000	0.99970	0.99930	1.00000	1.00000	1.00000	1.00000	1.00000	0.99930	0.99930
KICH	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
KIRC	1.00000	0.99040	0.99040	0.99040	0.99010	0.99040	0.99040	0.99040	0.99040	0.99040	0.98780	0.99040
KIRP	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
LIHC	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
LUAD	1.00000	1.00000	1.00000	1.00000	0.99500	1.00000	1.00000	1.00000	0.99930	1.00000	0.99830	0.99770
LUSC	1.00000	0.99100	0.99100	0.99100	0.99130	0.98960	0.99100	0.99060	0.99030	0.99100	0.99100	0.98990
PAAD	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
PCPG	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
READ	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
SARC	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
SKCM	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
STAD	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
THCA	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
THYM	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
UCEC	1.00000	0.99550	0.99640	0.98380	0.98200	0.99460	0.99100	0.99010	0.98470	0.99370	0.97840	0.97930
Ranking (W T L)	3 19 0	0 19 3	0 19 3	0 18 4	0 16 6	0 19 3	0 19 3	0 19 3	0 18 4	0 19 3	0 16 6	0 17 5

Boldface values denote the best results.

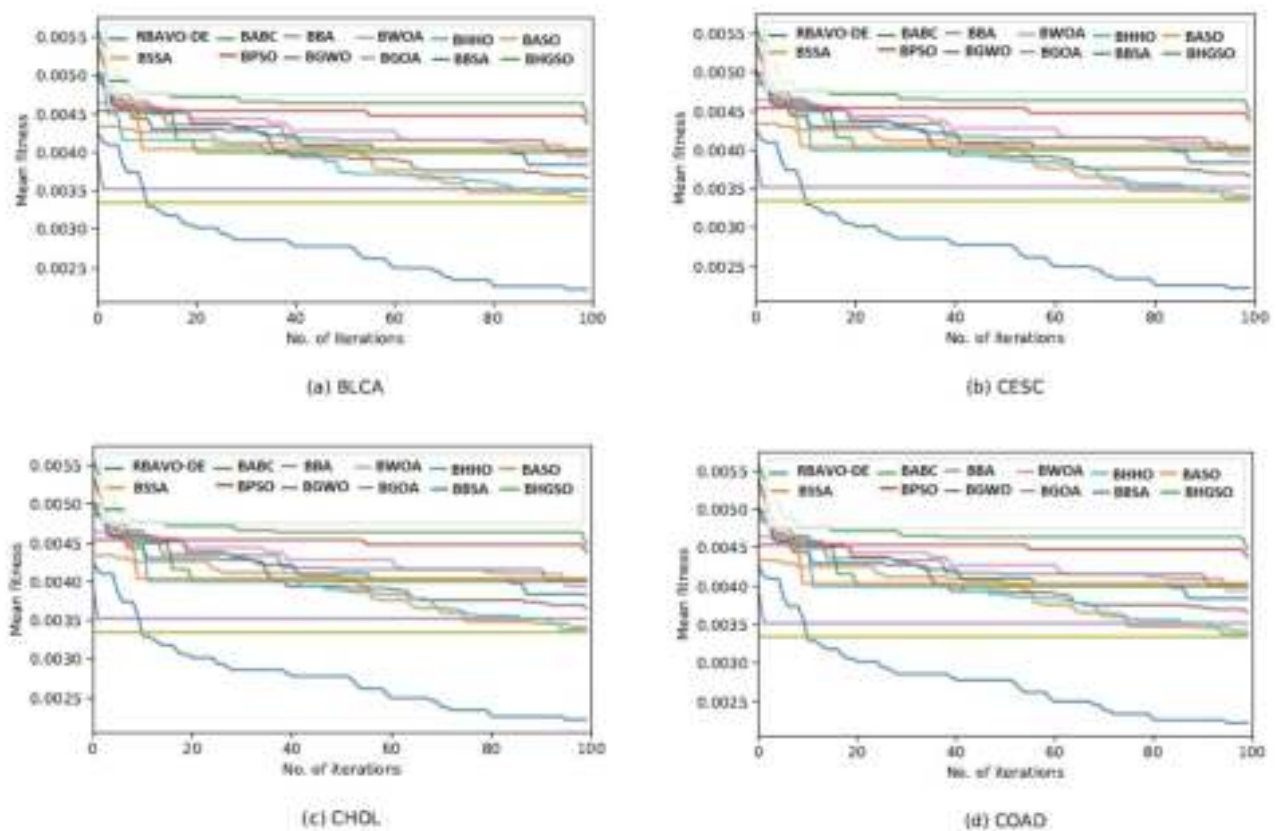
Table A10. The results of the proposed RBAVO-DE algorithm and its counterparts employing SVM regarding average F1-score values.

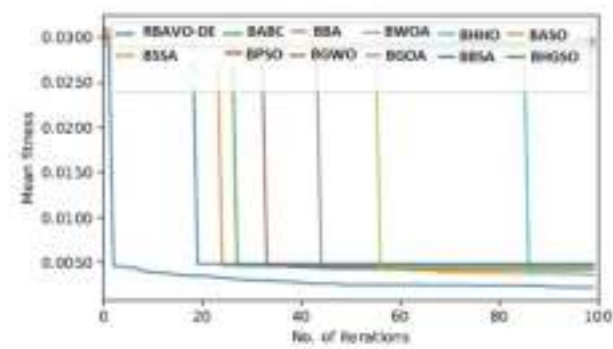
Dataset	RBAVO-DE	BSSA	BABC	BPSO	BBA	BGWO	BWOA	BGOA	BHHO	BBSA	BASO	BHGSO
BLCA	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
CESC	1.00000	1.00000	1.00000	1.00000	0.99950	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
CHOL	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
COAD	1.00000	1.00000	1.00000	1.00000	0.99910	1.00000	1.00000	1.00000	1.00000	1.00000	0.99970	1.00000
ESCA	1.00000	1.00000	1.00000	1.00000	0.99910	1.00000	1.00000	1.00000	1.00000	1.00000	0.99820	0.99600
GBM	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
HNSC	1.00000	0.96980	0.97000	0.96760	0.96250	0.96980	0.96900	0.96970	0.96970	0.96940	0.95730	0.96420
KICH	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
KIRC	1.00000	0.99520	0.99520	0.99520	0.99470	0.99520	0.99520	0.99520	0.99520	0.99520	0.99320	0.99520
KIRP	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
LIHC	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
LUAD	1.00000	1.00000	1.00000	1.00000	0.99690	1.00000	1.00000	1.00000	0.99970	1.00000	0.99880	0.99850
LUSC	1.00000	0.99480	0.99480	0.99480	0.99480	0.99480	0.99480	0.99480	0.99480	0.99480	0.99480	0.99480
PAAD	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
PCPG	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
READ	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
SARC	0.99900	0.99970	0.99810	0.99970	0.99750	0.99970	1.00000	0.99970	0.99810	0.99970	0.99710	0.99560
SKCM	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
STAD	0.99410	0.97350	0.97490	0.96990	0.96700	0.97290	0.97270	0.97270	0.96860	0.97250	0.96560	0.96640
THCA	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
THYM	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
UCEC	1.00000	0.99730	0.99730	0.99090	0.98730	0.99730	0.99450	0.99410	0.99180	0.99680	0.98680	0.98640
Ranking (W T L)	5 17 0	0 16 6	0 16 6	0 16 6	0 12 10	0 16 6	0 17 5	0 16 6	0 15 7	0 16 6	0 13 9	0 14 8

Boldface values denote the best results.

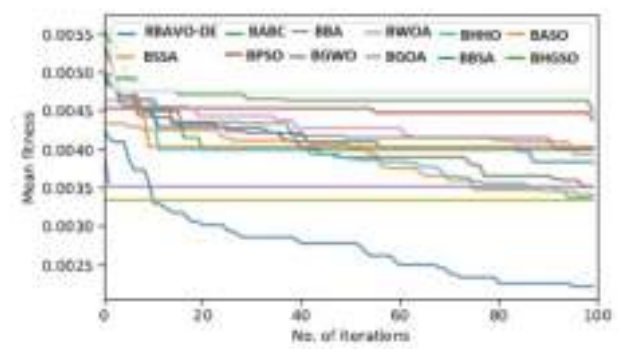
Appendix B. Convergence Graphs of the Proposed RBAVO-DE and Competitive Methods

Appendix B.1. Convergence Graphs Employing the SVM Model

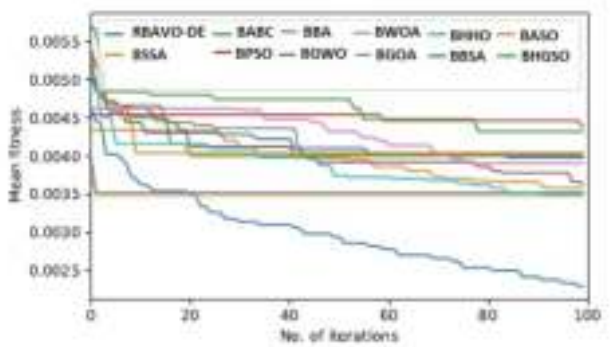
**Figure A1.** Cont.



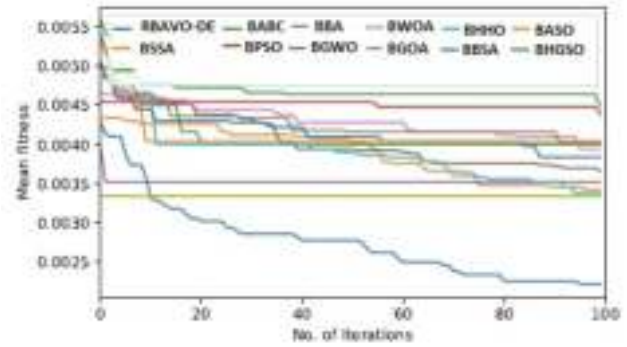
(e) ESCA



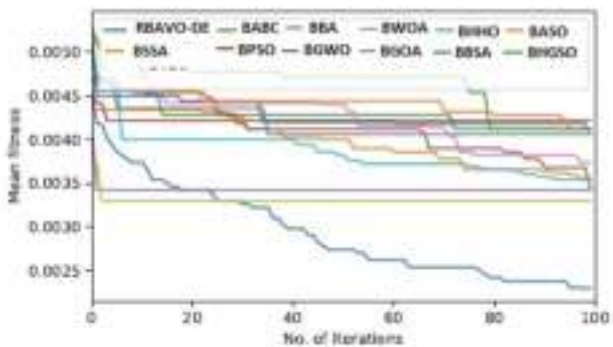
(f) GBM



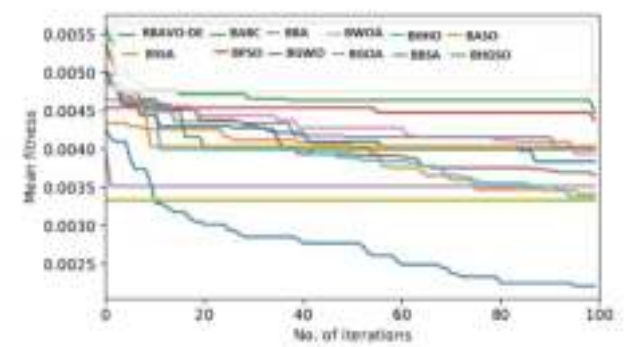
(g) HNSC



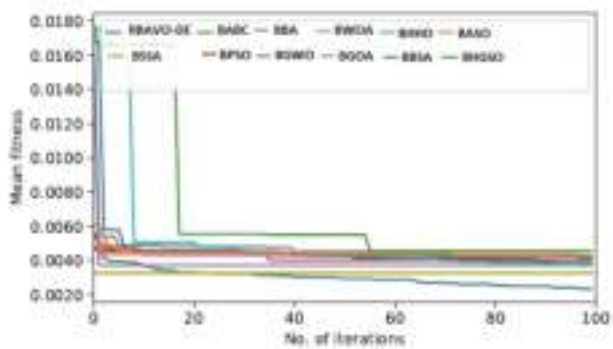
(h) KICH



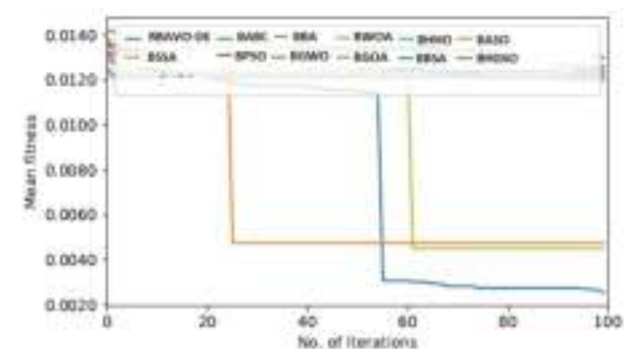
(i) KiRC



(j) KiRP



(k) LIHC



(l) LUAD

Figure A1. Cont.

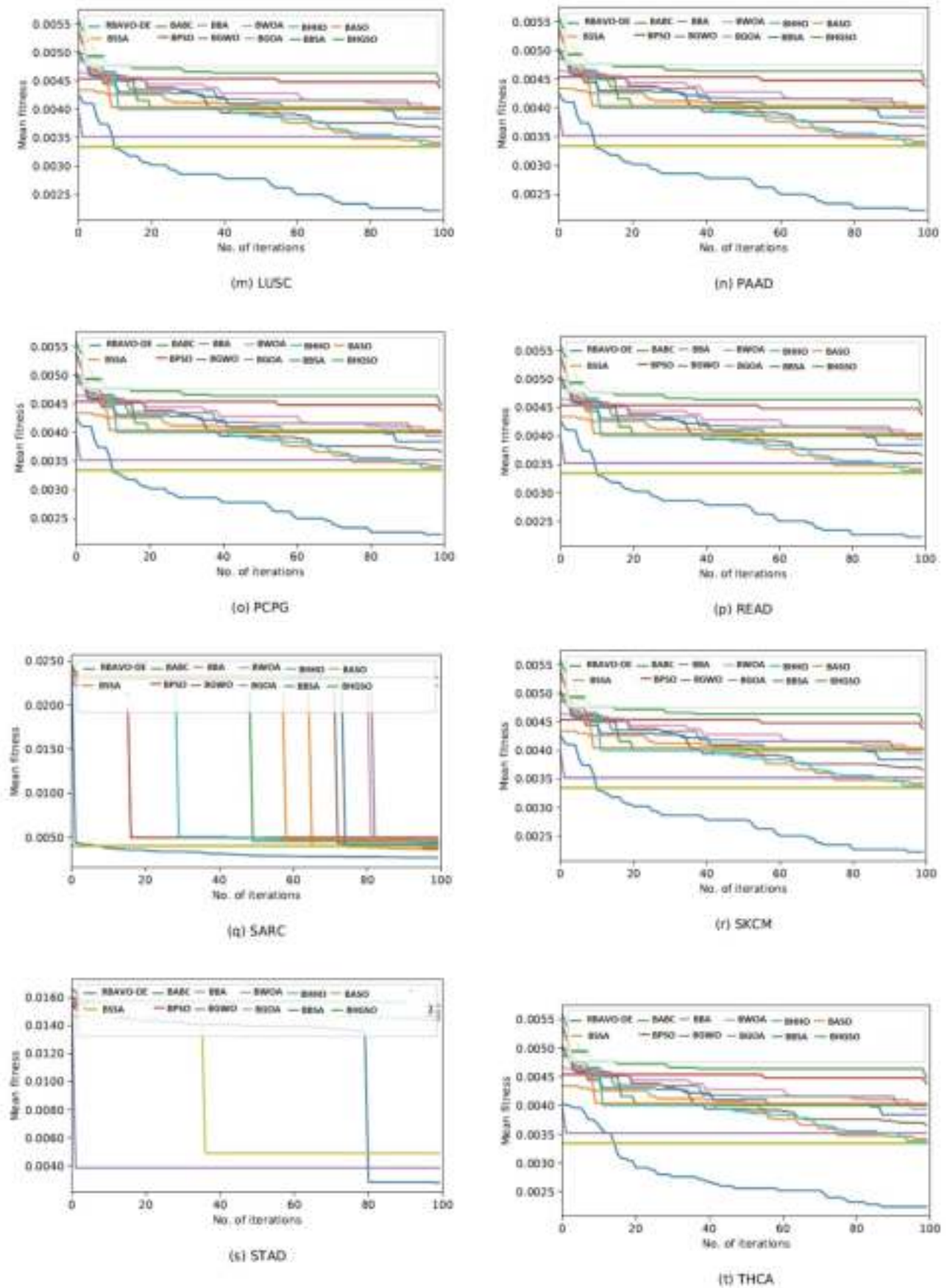


Figure A1. Cont.

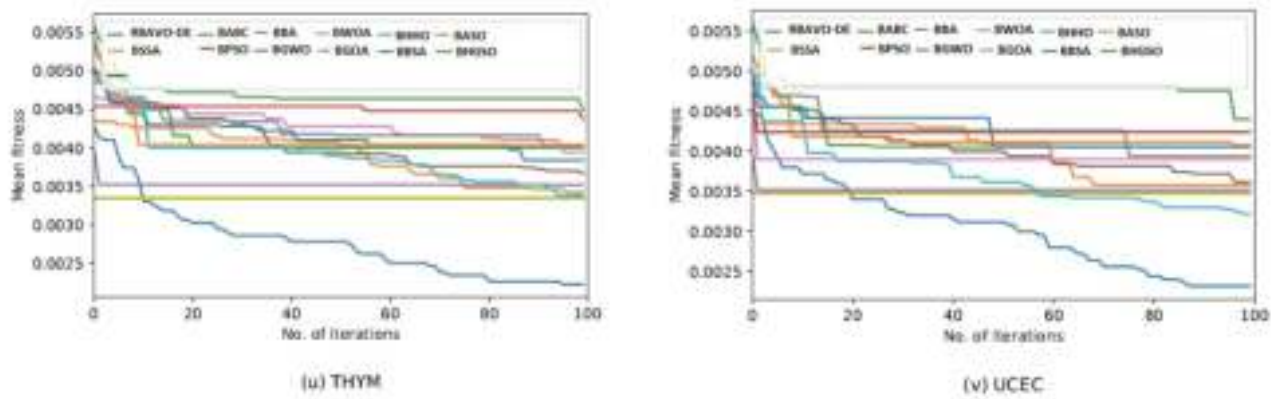


Figure A1. Convergence graphs of the proposed RBAVO-DE algorithm and competitive methods employing the SVM model on the overall benchmarks [57].

Appendix B.2. Convergence Graphs Employing the k-NN Model

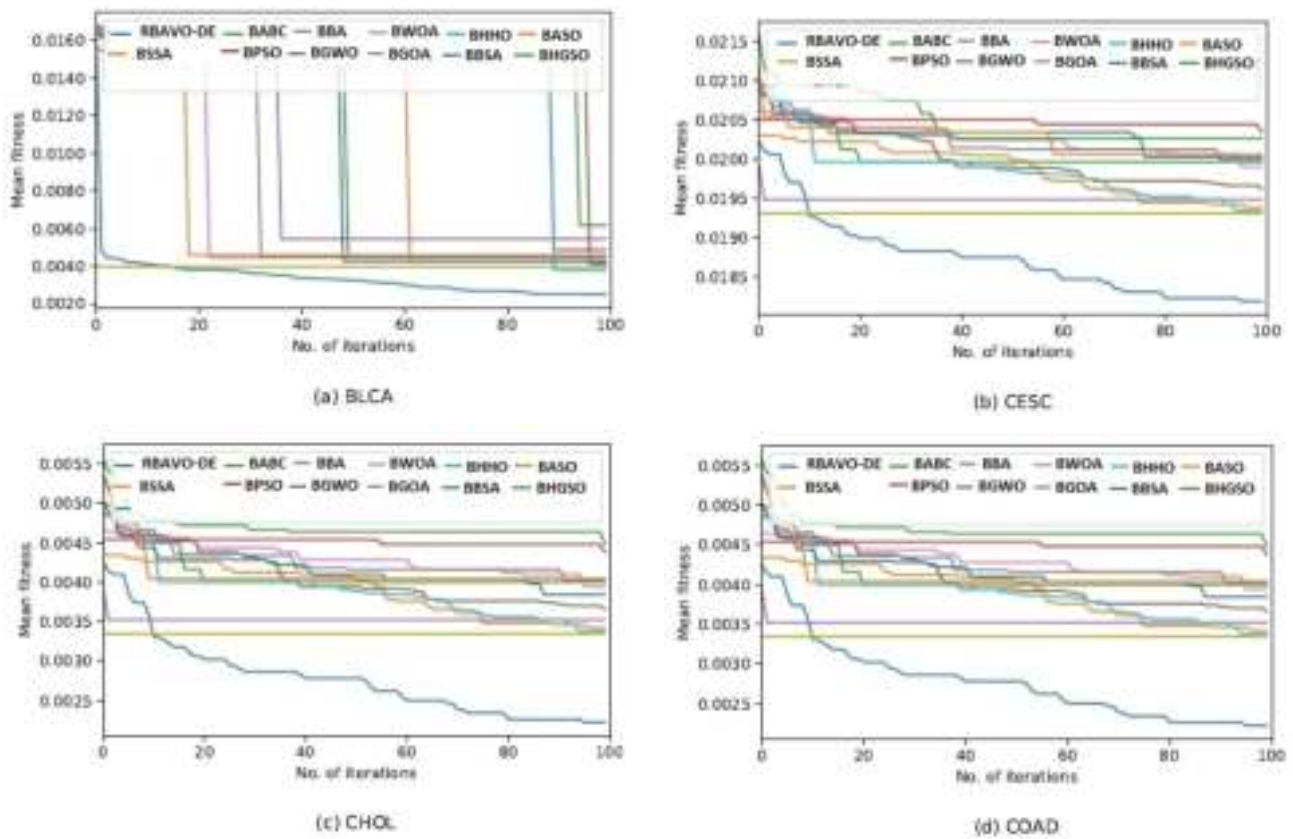
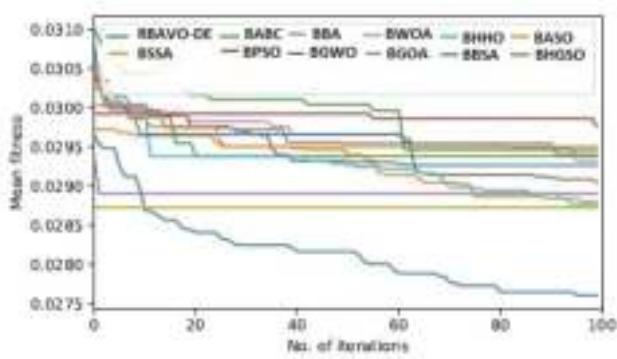
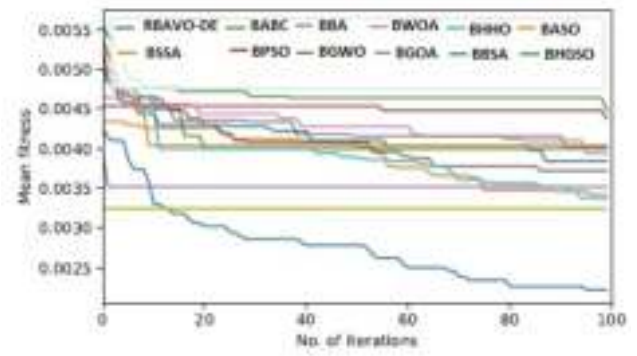


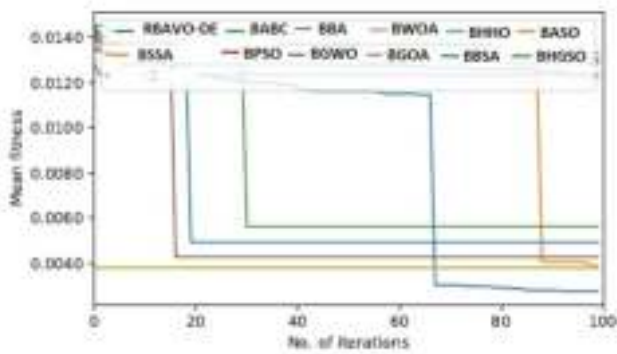
Figure A2. Cont.



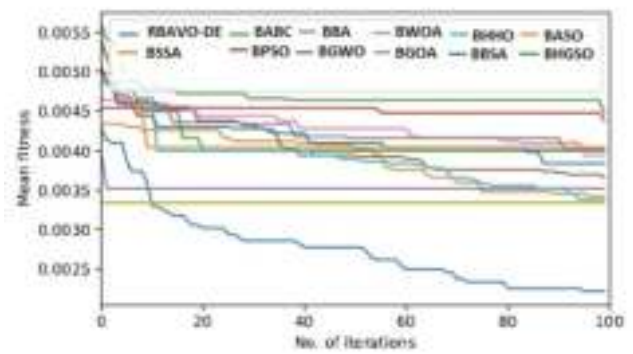
(e) ESCA



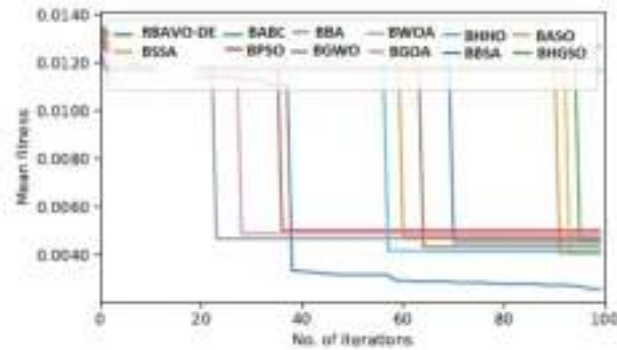
(f) GBM



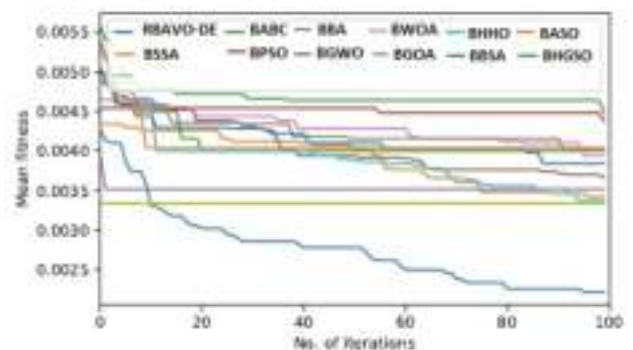
(g) HNSC



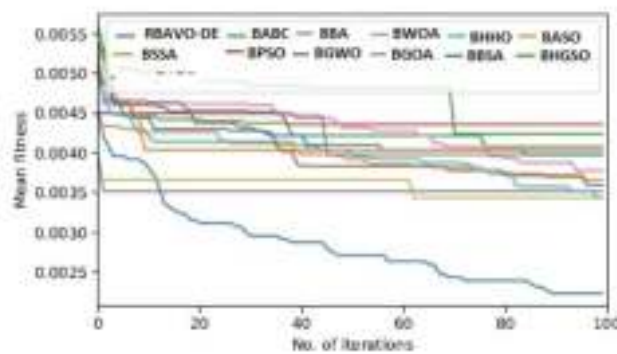
(h) KICHI



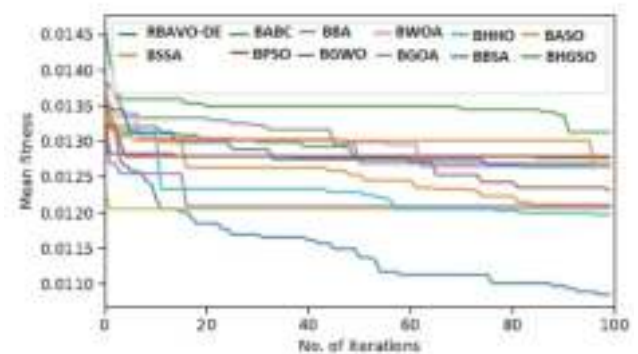
(i) KIRC



(j) KIRP



(k) LIHC



(l) LUAD

Figure A2. Cont.

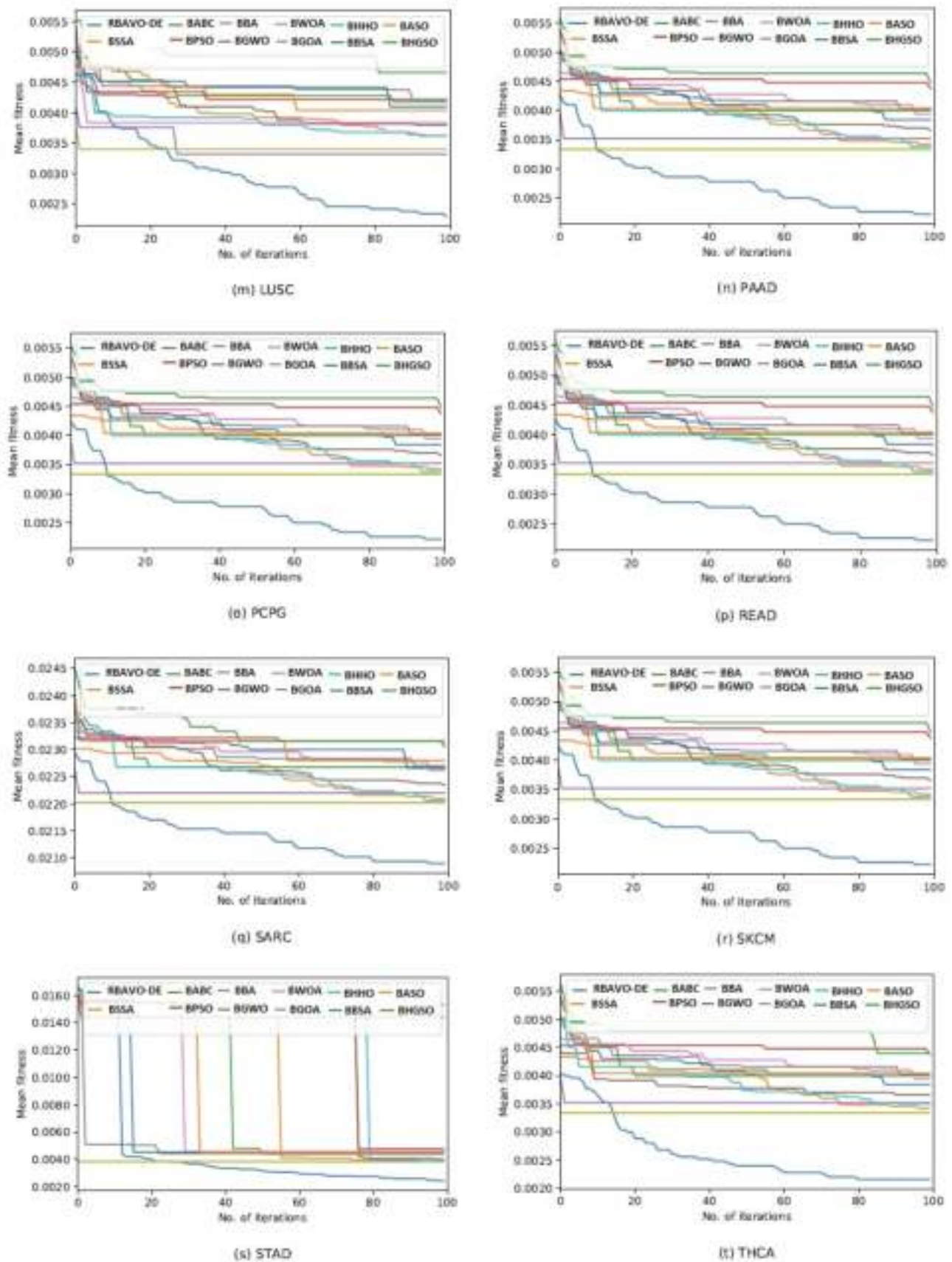


Figure A2. Cont.

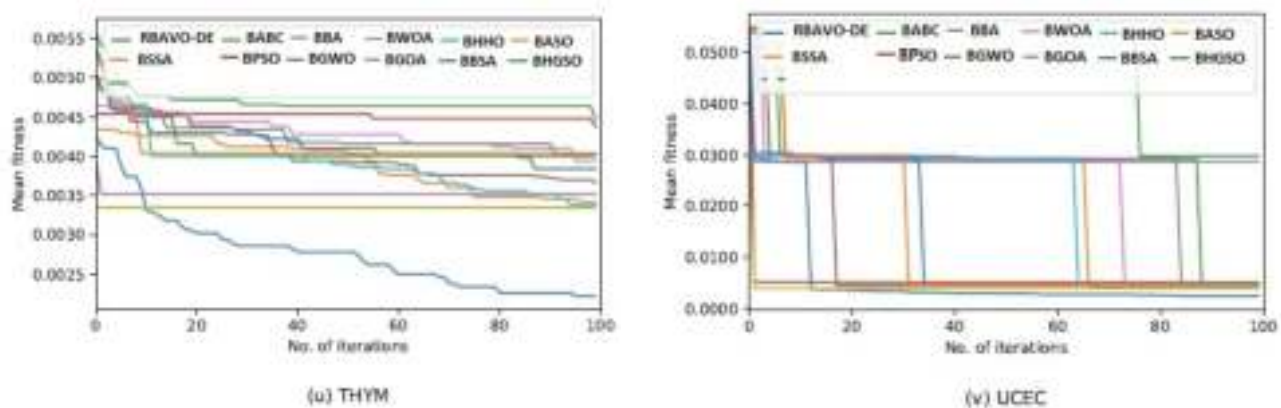


Figure A2. Convergence graphs of the proposed RBAVO-DE algorithm and competitive methods employing the k -NN model on the overall benchmarks (Cont.) [57].

References

- Estrada-Meza, C.; Torres-Copado, A.; Loreti González-Melgoza, L.; Ruiz-Manriquez, L.M.; De Donato, M.; Sharma, A.; Pathak, S.; Banerjee, A.; Paul, S. Recent insights into the microRNA and long non-coding RNA-mediated regulation of stem cell populations. *3 Biotech* **2022**, *12*, 270. [\[CrossRef\]](#) [\[PubMed\]](#)
- Kakati, T.; Bhattacharyya, D.K.; Kalita, J.K.; Norden-Krichmar, T.M. DEGnext: Classification of differentially expressed genes from RNA-seq data using a convolutional neural network with transfer learning. *BMC Bioinform.* **2022**, *23*, 17. [\[CrossRef\]](#)
- Zhao, S.; Fung-Leung, W.P.; Bittner, A.; Ngo, K.; Liu, X. Comparison of RNA-Seq and microarray in transcriptome profiling of activated T cells. *PLoS ONE* **2014**, *9*, e78644. [\[CrossRef\]](#)
- Chen, Z.; Luo, Z.; Zhang, D.; Li, H.; Liu, X.; Zhu, K.; Zhang, H.; Wang, Z.; Zhou, P.; Ren, J.; et al. TIGER: A web portal of tumor immunotherapy gene expression resource. *Genom. Proteom. Bioinform.* **2023**, *21*, 337–348. [\[CrossRef\]](#) [\[PubMed\]](#)
- Nunez-Garcia, J.; AbuOun, M.; Storey, N.; Brouwer, M.; Delgado-Blas, J.; Mo, S.S.; Ellaby, N.; Veldman, K.; Haenni, M.; Châtre, P.; et al. Harmonisation of in-silico next-generation sequencing based methods for diagnostics and surveillance. *Sci. Rep.* **2022**, *12*, 14372. [\[CrossRef\]](#)
- Wang, Z.; Gerstein, M.; Snyder, M. RNA-Seq: A revolutionary tool for transcriptomics. *Nat. Rev. Genet.* **2009**, *10*, 57–63. [\[CrossRef\]](#)
- Kim, W.J.; Choi, B.R.; Noh, J.J.; Lee, Y.Y.; Kim, T.J.; Lee, J.W.; Kim, B.G.; Choi, C.H. Comparison of RNA-Seq and microarray in the prediction of protein expression and survival prediction. *Front. Genet.* **2024**, *15*, 1342021. [\[CrossRef\]](#)
- Wang, M.; Chen, X.; Dai, Y.; Wu, D.; Liu, F.; Yang, Z.; Song, B.; Xie, L.; Yang, L.; Zhao, W.; et al. Concordance study of a 520-gene next-generation sequencing-based genomic profiling assay of tissue and plasma samples. *Mol. Diagn. Ther.* **2022**, *26*, 309–322. [\[CrossRef\]](#)
- Metzker, M.L. Sequencing technologies—The next generation. *Nat. Rev. Genet.* **2010**, *11*, 31–46. [\[CrossRef\]](#) [\[PubMed\]](#)
- Pandey, D.; Onkara Perumal, P. A scoping review on deep learning for next-generation RNA-Seq. data analysis. *Funct. Integr. Genom.* **2023**, *23*, 134. [\[CrossRef\]](#)
- Liu, S.; Yao, W. Prediction of lung cancer using gene expression and deep learning with KL divergence gene selection. *BMC Bioinform.* **2022**, *23*, 175. [\[CrossRef\]](#) [\[PubMed\]](#)
- Houssein, E.H.; Oliva, D.; Celik, E.; Emam, M.M.; Ghoniem, R.M. Boosted sooty tern optimization algorithm for global optimization and feature selection. *Expert Syst. Appl.* **2023**, *213*, 119015. [\[CrossRef\]](#)
- Joshi, A.A.; Aziz, R.M. A two-phase cuckoo search based approach for gene selection and deep learning classification of cancer disease using gene expression data with a novel fitness function. *Multimed. Tools Appl.* **2024**, *83*, 71721–71752. [\[CrossRef\]](#)
- Ramaswamy, R.; Kandhasamy, P.; Palaniswamy, S. Feature selection for Alzheimer's gene expression data using modified binary particle swarm optimization. *IETE J. Res.* **2023**, *69*, 9–20. [\[CrossRef\]](#)
- Cui, X.; Li, Y.; Fan, J.; Wang, T. A novel filter feature selection algorithm based on relief. *Appl. Intell.* **2022**, *52*, 5063–5081. [\[CrossRef\]](#)
- Alhenawi, E.; Al-Sayyed, R.; Hudaib, A.; Mirjalili, S. Feature selection methods on gene expression microarray data for cancer classification: A systematic review. *Comput. Biol. Med.* **2022**, *140*, 105051. [\[CrossRef\]](#) [\[PubMed\]](#)
- Parlak, B.; Uysal, A.K. A novel filter feature selection method for text classification: Extensive Feature Selector. *J. Inf. Sci.* **2023**, *49*, 59–78. [\[CrossRef\]](#)
- Albulayhi, K.; Abu Al-Haija, Q.; Alsuhbany, S.A.; Jillepalli, A.A.; Ashrafuzzaman, M.; Sheldon, F.T. IoT intrusion detection using machine learning with a novel high performing feature selection method. *Appl. Sci.* **2022**, *12*, 5015. [\[CrossRef\]](#)
- Fatima, A.; Nazir, T.; Nazir, A.K.; Din, A.M.U. An efficient Incremental Wrapper-based Information Gain Gene Subset Selection (IG based on IWSSr) method for Tumor Discernment. *Multimed. Tools Appl.* **2024**, *83*, 64741–64766. [\[CrossRef\]](#)
- Kaur, S.; Kumar, Y.; Koul, A.; Kumar Kamboj, S. A systematic review on metaheuristic optimization techniques for feature selections in disease diagnosis: Open issues and challenges. *Arch. Comput. Methods Eng.* **2023**, *30*, 1863–1895. [\[CrossRef\]](#)

21. Abd El-Mageed, A.A.; Gad, A.G.; Sallam, K.M.; Munasinghe, K.; Abohany, A.A. Improved Binary Adaptive Wind Driven Optimization Algorithm-Based Dimensionality Reduction for Supervised Classification. *Comput. Ind. Eng.* **2022**, *167*, 107904. [\[CrossRef\]](#)
22. Gad, A.G.; Sallam, K.M.; Chakraborty, R.K.; Ryan, M.J.; Abohany, A.A. An improved binary sparrow search algorithm for feature selection in data classification. *Neural Comput. Appl.* **2022**, *34*, 15705–15752. [\[CrossRef\]](#)
23. Hussien, R.M.; Abohany, A.A.; Abd El-Mageed, A.A.; Hosny, K.M. Improved Binary Meerkat Optimization Algorithm for efficient feature selection of supervised learning classification. *Knowl.-Based Syst.* **2024**, *292*, 111616. [\[CrossRef\]](#)
24. Abd El-Mageed, A.A.; Abohany, A.A.; Elashry, A. Effective Feature Selection Strategy for Supervised Classification based on an Improved Binary Aquila Optimization Algorithm. *Comput. Ind. Eng.* **2023**, *181*, 109300. [\[CrossRef\]](#)
25. Yin, Y.; Jang-Jaccard, J.; Xu, W.; Singh, A.; Zhu, J.; Sabrina, F.; Kwak, J. IGRF-RFE: A hybrid feature selection method for MLP-based network intrusion detection on UNSW-NB15 dataset. *J. Big Data* **2023**, *10*, 15. [\[CrossRef\]](#)
26. Nakao, H.; Imaoka, M.; Hida, M.; Imai, R.; Nakamura, M.; Matsumoto, K.; Kita, K. Determination of individual factors associated with hallux valgus using SVM-RFE. *BMC Musculoskelet. Disord.* **2023**, *24*, 534. [\[CrossRef\]](#) [\[PubMed\]](#)
27. Sarafrazi, S.; Nezamabadi-Pour, H. Facing the classification of binary problems with a GSA-SVM hybrid system. *Math. Comput. Model.* **2013**, *57*, 270–278. [\[CrossRef\]](#)
28. Cadenas, J.M.; Garrido, M.C.; MartíNez, R. Feature subset selection filter–wrapper based on low quality data. *Expert Syst. Appl.* **2013**, *40*, 6241–6252. [\[CrossRef\]](#)
29. Oh, I.S.; Lee, J.S.; Moon, B.R. Hybrid genetic algorithms for feature selection. *IEEE Trans. Pattern Anal. Mach. Intell.* **2004**, *26*, 1424–1437.
30. Abdollahzadeh, B.; Gharehchopogh, F.S.; Mirjalili, S. African vultures optimization algorithm: A new nature-inspired metaheuristic algorithm for global optimization problems. *Comput. Ind. Eng.* **2021**, *158*, 107408. [\[CrossRef\]](#)
31. El-Shafeiy, E.; Hassanien, A.E.; Sallam, K.M.; Abohany, A. Approach for training quantum neural network to predict severity of COVID-19 in patients. *Comput. Mater. Contin.* **2020**, *66*, 1745–1755. [\[CrossRef\]](#)
32. Yaqoob, A.; Verma, N.K.; Aziz, R.M. Optimizing gene selection and cancer classification with hybrid sine cosine and cuckoo search algorithm. *J. Med. Syst.* **2024**, *48*, 10. [\[CrossRef\]](#)
33. Joshi, A.A.; Aziz, R.M. Deep learning approach for brain tumor classification using metaheuristic optimization with gene expression data. *Int. J. Imaging Syst. Technol.* **2023**, *34*, e23007. [\[CrossRef\]](#)
34. Mahto, R.; Ahmed, S.U.; Rahman, R.U.; Aziz, R.M.; Roy, P.; Mallik, S.; Li, A.; Shah, M.A. A novel and innovative cancer classification framework through a consecutive utilization of hybrid feature selection. *BMC Bioinform.* **2023**, *24*, 479. [\[CrossRef\]](#) [\[PubMed\]](#)
35. Neggaz, N.; Neggaz, I.; Abd Elaziz, M.; Hussien, A.G.; Abulaigh, L.; Damaševičius, R.; Hu, G. Boosting manta rays foraging optimizer by trigonometry operators: A case study on medical dataset. *Neural Comput. Appl.* **2024**, *36*, 9405–9436. [\[CrossRef\]](#)
36. Lyu, B.; Haque, A. Deep learning based tumor type classification using gene expression data. In Proceedings of the 2018 ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics, Washington, DC, USA, 29 August–1 September 2018; pp. 89–96.
37. Khalifa, N.E.M.; Taha, M.H.N.; Ali, D.E.; Slowik, A.; Hassanien, A.E. Artificial intelligence technique for gene expression by tumor RNA-Seq data: A novel optimized deep learning approach. *IEEE Access* **2020**, *8*, 22874–22883. [\[CrossRef\]](#)
38. Dillies, M.A.; Rau, A.; Aubert, J.; Hennequet-Antier, C.; Jeanmougin, M.; Servant, N.; Keime, C.; Marot, G.; Castel, D.; Estelle, J.; et al. A comprehensive evaluation of normalization methods for Illumina high-throughput RNA sequencing data analysis. *Briefings Bioinform.* **2013**, *14*, 671–683. [\[CrossRef\]](#)
39. Xiao, Y.; Wu, J.; Lin, Z.; Zhao, X. A deep learning-based multi-model ensemble method for cancer prediction. *Comput. Methods Programs Biomed.* **2018**, *153*, 1–9. [\[CrossRef\]](#)
40. Liu, M.; Xu, L.; Yi, J.; Huang, J. A feature gene selection method based on ReliefF and PSO. In Proceedings of the 2018 10th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA), Changsha, China, 7–8 January 2018; IEEE: New York, NY, USA, 2018; pp. 298–301.
41. Kononenko, I. Estimating attributes: Analysis and extensions of RELIEF. In Proceedings of the European Conference on Machine Learning, Catania, Italy, 6–8 April 1994; Springer: Berlin/Heidelberg, Germany, 1994; pp. 171–182.
42. Faris, H.; Mafarja, M.M.; Heidari, A.A.; Aljarah, I.; Ala'M, A.Z.; Mirjalili, S.; Fujita, H. An efficient binary salp swarm algorithm with crossover scheme for feature selection problems. *Knowl.-Based Syst.* **2018**, *154*, 43–67. [\[CrossRef\]](#)
43. Abdel-Basset, M.; Ding, W.; El-Shahat, D. A hybrid Harris Hawks optimization algorithm with simulated annealing for feature selection. *Artif. Intell. Rev.* **2020**, *54*, 593–637. [\[CrossRef\]](#)
44. Storn, R.; Price, K. Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [\[CrossRef\]](#)
45. Karaboga, D.; Basturk, B. A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. *J. Glob. Optim.* **2007**, *39*, 459–471. [\[CrossRef\]](#)
46. Mirjalili, S.; Gandomi, A.H.; Mirjalili, S.Z.; Saremi, S.; Faris, H.; Mirjalili, S.M. Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems. *Adv. Eng. Softw.* **2017**, *114*, 163–191. [\[CrossRef\]](#)
47. Khanesar, M.A.; Teshnehlab, M.; Shoorehdeli, M.A. A novel binary particle swarm optimization. In Proceedings of the 2007 Mediterranean Conference on Control & Automation, Athens, Greece, 27–29 June 2007; IEEE: New York, NY, USA, 2007; pp. 1–6.

48. Mirjalili, S.; Mirjalili, S.M.; Yang, X.S. Binary bat algorithm. *Neural Comput. Appl.* **2014**, *25*, 663–681. [\[CrossRef\]](#)
49. Emary, E.; Zawbaa, H.M.; Hassanien, A.E. Binary grey wolf optimization approaches for feature selection. *Neurocomputing* **2016**, *172*, 371–381. [\[CrossRef\]](#)
50. Hichem, H.; Elkamel, M.; Rafik, M.; Mesaaoud, M.T.; Ouahiba, C. A new binary grasshopper optimization algorithm for feature selection problem. *J. King Saud Univ. Comput. Inf. Sci.* **2022**, *34*, 316–328. [\[CrossRef\]](#)
51. Hussien, A.G.; Hassanien, A.E.; Houssein, E.H.; Bhattacharyya, S.; Amin, M. S-shaped binary whale optimization algorithm for feature selection. In *Recent Trends in Signal and Image Processing: ISSIP 2017*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 79–87.
52. Zhao, W.; Wang, L.; Zhang, Z. Atom search optimization and its application to solve a hydrogeologic parameter estimation problem. *Knowl.-Based Syst.* **2019**, *163*, 283–304. [\[CrossRef\]](#)
53. Meng, X.B.; Gao, X.Z.; Lu, L.; Liu, Y.; Zhang, H. A new bio-inspired optimisation algorithm: Bird Swarm Algorithm. *J. Exp. Theor. Artif. Intell.* **2016**, *28*, 673–687. [\[CrossRef\]](#)
54. Hashim, F.A.; Houssein, E.H.; Mabrouk, M.S.; Al-Atabany, W.; Mirjalili, S. Henry gas solubility optimization: A novel physics-based algorithm. *Future Gener. Comput. Syst.* **2019**, *101*, 646–667. [\[CrossRef\]](#)
55. Heidari, A.A.; Mirjalili, S.; Faris, H.; Aljarah, I.; Mafarja, M.; Chen, H. Harris hawks optimization: Algorithm and applications. *Future Gener. Comput. Syst.* **2019**, *97*, 849–872. [\[CrossRef\]](#)
56. Normalized-level3 RNA-Seq Gene Expression Dataset. Available online: <https://gdac.broadinstitute.org/> (accessed on 20 December 2023).
57. El-Mageed, A.A.A.; Elkhoul, A.E.; Abohany, A.A.; Gafar, M. Gene selection via improved nuclear reaction optimization algorithm for cancer classification in high-dimensional data. *J. Big Data* **2024**, *11*, 46. [\[CrossRef\]](#)
58. Mafarja, M.; Mirjalili, S. Whale optimization approaches for wrapper feature selection. *Appl. Soft Comput.* **2018**, *62*, 441–453. [\[CrossRef\]](#)
59. Thaher, T.; Heidari, A.A.; Mafarja, M.; Dong, J.S.; Mirjalili, S. Binary Harris Hawks optimizer for high-dimensional, low sample size feature selection. In *Evolutionary Machine Learning Techniques*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 251–272.
60. Derrac, J.; García, S.; Molina, D.; Herrera, F. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm Evol. Comput.* **2011**, *1*, 3–18. [\[CrossRef\]](#)
61. Python Code for Gene Selection via Relief Binary African Vultures Optimization Integrated with Differential Evolution. Available online: https://github.com/D-Amr-Atef/Gene_Selection_RBAVO-DE.git (accessed on 18 May 2024).

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.



A Review on Reinforcement Learning in Production Scheduling: An Inferential Perspective

Vladimir Modrak ^{1,*} , Ranjitharamasamy Sudhakarapandian ² , Arunmozhi Balamurugan ²
and Zuzana Soltysova ¹

¹ Faculty of Manufacturing Technologies, Technical University of Kosice, 080 01 Prešov, Slovakia; zuzana.soltysova@tuke.sk

² School of Mechanical Engineering, Vellore Institute of Technology, Vellore 632014, Tamil Nadu, India; sudhakarapandian.r@vit.ac.in (R.S.); arunmozhi.b@vit.ac.in (A.B.)

* Correspondence: vladimir.modrak@tuke.sk

Abstract: In this study, a systematic review on production scheduling based on reinforcement learning (RL) techniques using especially bibliometric analysis has been carried out. The aim of this work is, among other things, to point out the growing interest in this domain and to outline the influence of RL as a type of machine learning on production scheduling. To achieve this, the paper explores production scheduling using RL by investigating the descriptive metadata of pertinent publications contained in Scopus, ScienceDirect, and Google Scholar databases. The study focuses on a wide spectrum of publications spanning the years between 1996 and 2024. The findings of this study can serve as new insights for future research endeavors in the realm of production scheduling using RL techniques.

Keywords: bibliometric analysis; production scheduling; reinforcement learning



Citation: Modrak, V.; Sudhakarapandian, R.; Balamurugan, A.; Soltysova, Z. A Review on Reinforcement Learning in Production Scheduling: An Inferential Perspective. *Algorithms* **2024**, *17*, 343. <https://doi.org/10.3390/a17080343>

Academic Editors: Alexandre Dolgui, David Lemoine, María I. Restrepo and Frank Werner

Received: 6 June 2024

Revised: 28 July 2024

Accepted: 2 August 2024

Published: 7 August 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Production scheduling is considered as one of the most critical elements of manufacturing management in aligning production activities with business objectives, in ensuring a smooth flow of goods resources, and in supporting company's ability to remain competitive in the marketplace. Scheduling algorithms play an important role in enhancing production efficiency and effectiveness, and therefore have long been a subject of extensive research in various interdisciplinary domains, such as industrial engineering, automation, and management science [1]. The production scheduling tasks can be solved using three main types of step-by-step procedures such as exact algorithms, heuristic algorithms, and meta-heuristic algorithms [2,3]. Although an exact algorithm can theoretically guarantee the optimum solution, the NP-hardness of major problems makes them impossible to address effectively and efficiently [4]. Heuristics use a set of rules to create scheduling solutions quickly and effectively without consideration of global optimization. Furthermore, the creation of rules is heavily reliant on a thorough comprehension of the particulars of the situation [5–8]. Whereas meta-heuristics can produce good scheduling solutions in a reasonable amount of computing time, the way search operators create them significantly depends on the specific situation at hand [9–13]. In addition, the iterative search process poses challenges in terms of time consumption and applicability in real-time scenarios when dealing with large-scale problems. Scheduling approaches based on reinforcement learning (RL) have proven to be a useful tool in this regard. Reinforcement learning is a subfield within the broader domain of machine learning. RL is considered one of the most perspective approaches for robust cooperative scheduling, which allows production managers to interact with a complex manufacturing environment, learn from previous experience, and select optimal decisions. It involves the process of an agent autonomously selecting and executing actions to accomplish a given task. The agent learns via experience and aims to maximize the

rewards it receives in certain scenarios. The primary goal of RL is to optimize the cumulative reward obtained by an agent through the evaluation and selection of actions within a dynamic environment [14–20]. The most current development in artificial intelligence technology has allowed successful application of RL in sequential decision-making problems with multiple objectives which are usable for robot scheduling and control [21,22]. The research on production scheduling using RL since 1998 has been evolving as advancing optimization techniques compared to metaheuristics. RL significantly improves the computational efficiency of addressing scheduling problems. Numerous research articles of RL on production scheduling have been undertaken since its inception in 1996, establishing a substantial and valuable foundation (see few of them, e.g., [23–27]), but only two recent review papers [28,29] paid attention to this subject. However, both the papers published in 2021 were focused on different perspectives of RL-based scheduling algorithms than presented in this paper. In other words, the content of the proposed paper is in a disjunctive and complementary relation to the two mentioned studies.

The research question of this paper is multi-faceted including multiple features of this domain that require separate answers. The main features that this paper will address are: What are the main emerging research areas in the field? Which related topics are being covered in production scheduling based on RL? What are typical implementation domains within RL applied to production scheduling?

Its novelty lies in providing the additional analyses and assessments regarding RL in production scheduling. It encompasses, e.g., the citation trend for RL on production scheduling, the most influential authors in this domain, the most relevant sources in the field, comparison of deterministic types of scheduling methods and uncertain types of scheduling methods.

2. Materials and Methods

The bibliometric approach employed here, as the main research method, presents a quantitative instrument for monitoring and representing scientific progress by examining and visualizing scientific information. The growing acceptance of bibliometric methods in several academic disciplines indicates that their use brings expected effects [30–33]. The present investigation of RL-based production scheduling is conducted utilizing the procedure comprising of five coherent phases, as depicted in Figure 1.

Moreover, this research follows an inferential approach, where the sample of population is explored to determine its characteristics [34,35]. Moreover, the inferential concept of scientific representation proposed by Suárez [36] was applied here to formulate research outputs. Its essence is to employ alternative reasoning to reach results that differ from the isomorphic view of scientific representation in the sense that empirical knowledge plays an important role in inductive reasoning [37,38].

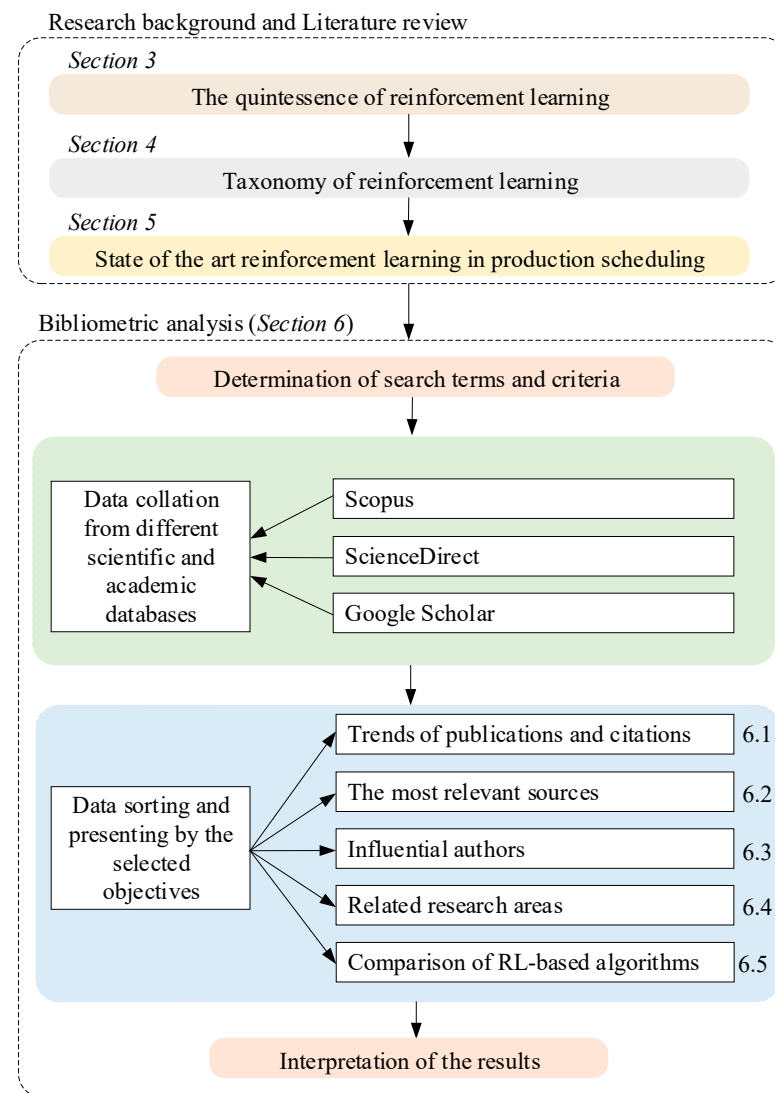


Figure 1. Research methodology framework.

3. The Quintessence of Reinforcement Learning

Artificial Intelligence (AI) has become an integral and significant aspect of our daily life as we approach the conclusion of this decade. Artificial Intelligence has been present for the past seven decades but has gained significant momentum in the recent three decades, leading to extensive research in this field [39]. AI refers to various techniques that allow computers to acquire knowledge and make choices by analyzing data. Out of these techniques, machine learning (ML) has made significant advancements in the past two decades, transitioning from a mere curiosity in laboratories to a widely used practical technology in commercial applications. In AI, ML has become the preferred approach for creating functional software in areas such as computer vision, natural language processing, speech recognition, robot control, and various other applications [40].

Many AI developers now acknowledge that, in several cases, it is more convenient to train a system by demonstrating ideal input-output behavior rather than manually programming it to anticipate the desired response for every potential input. ML has significantly impacted various areas of computer science and sectors that deal with data-driven problems. It includes consumer services, diagnosing defects in intricate systems, and managing logistics chains. ML approaches have had diverse effects on several empirical sciences, such as biology, cosmology, and social science. These approaches have been used creatively to examine vast experimental data [41].

Three major ML paradigms are supervised learning, unsupervised learning, and reinforcement learning [15]. Supervised learning entails the process of training a model using a dataset that has been labeled, meaning that each input is associated with its corresponding accurate output. The objective is to acquire knowledge about a transformation from given inputs to corresponding outputs, which can then be utilized to make accurate forecasts on novel, unobserved data. Unsupervised learning involves the identification of concealed patterns or inherent structures in unlabeled input data. The model aims to acquire knowledge about the fundamental distribution or grouping of the data without explicit instructions on the desired outcome [28]. RL is centered around teaching an agent to make numerous choices by engaging with an environment and receiving rewards or punishments as feedback. The agent acquires the ability to optimize its activities to maximize the total rewards it receives over a period [42].

Due to its dynamic and adaptive methodology, RL is highly effective for addressing problems in the manufacturing systems and robotics research domain [22,28]. RL is distinct from supervised learning, where a model is trained on a predetermined dataset, and unsupervised learning, where the model discovers concealed patterns within data. In reinforcement learning, an agent is educated to make a series of decisions by engaging with an environment. The agent's objective is to acquire a strategy, sometimes known as a policy, that optimizes the total rewards obtained over time [43].

RL is a decision-making process influenced by behavioral psychology. It involves an agent learning to achieve a goal by acting and receiving feedback from the environment through rewards or penalties. This trial-and-error method is akin to how humans and animals learn from their environment. The critical components of RL are the agent, the environment, actions, states, and rewards. The agent observes the environment, takes actions, and receives rewards, which are used to guide future actions [14]. RL is particularly effective in tasks with uncertain outcomes, such as playing games, controlling robots, and managing resources. In these situations, the agent must balance exploration (trying new actions) and exploitation (using established actions that lead to significant rewards) to develop an optimal policy efficiently [42].

The mathematical framework of RL is commonly represented using Markov Decision Processes (MDPs), which offer a systematic approach to modeling decision-making problems, including both random and agent-controlled outcomes. Solving an MDP entails determining a policy that prescribes the optimal action to be taken in each stage, aiming to maximize the predicted cumulative rewards in the future. The recent progress in deep learning has dramatically improved the capacities of RL [15,16]. The combination of deep neural networks and RL algorithms has resulted in the emergence of deep reinforcement learning (DRL), which has demonstrated exceptional achievements in intricate situations. Prominent instances include AlphaGo, developed by Google, which triumphed over human champions in the game of Go, as well as diverse implementations in autonomous driving, where DRL algorithms acquire the ability to navigate intricate and ever-changing surroundings successfully. Reinforcement learning remains a dynamic and swiftly advancing domain in machine learning, where ongoing research constantly pushes the limits of what artificial agents can accomplish. As we delve into the complexities of RL, we discover fresh opportunities for intelligent systems that possess the ability to acquire knowledge and adjust their behavior in real-time. RL paves the way for a future where computers can seamlessly cooperate with people and function autonomously in more advanced manners [42].

4. Taxonomy of Reinforcement Learning Algorithms

Sutton and Barto [42] emphasize the crucial role of the three essential components in the RL process:

- A policy determines the actions to be taken in each environmental state.
- A reward signal categorizes these actions as beneficial or detrimental based on the immediate outcome of transitioning between states.

- A value function assesses the long-term effectiveness of actions by considering not only a state's immediate reward but also the anticipated future rewards.

The primary objective of the agent is to optimize the cumulative reward. Consequently, the reward signal is the foundation for modifying the policy and the value function. In specific RL systems, an additional component known as a model of the environment exists, which is optional. It replicates the behaviour of the climate or enables more comprehensive assumptions about the behaviour of the environment. Models are utilized for strategic planning, which involves making decisions based on anticipation of future scenarios before their actual occurrence [24]. Model-based methods are used to handle reinforcement learning problems by utilizing models and planning, in contrast to more straightforward model-free methods. They exhibit apparent trial-and-error learning behaviour and are considered the reverse of planning [26,27]. RL processes can be represented and analysed using a mathematical framework called MDP. The stochastic mathematical model can be formalized using a 5-tuple (s, a, p, r, γ) , in which the first symbol 's' represents the finite collection of all possible environment states, while s_t is the state at a specific time 't'. Letter 'a' represents the set of all possible actions, while the action taken at time 't' is marked as a_t . Symbol 'p' refers to the transition probabilities matrix, which defines the conditional probability of transitioning to a new state 's', with a reward 'r', given the current state 's', and action 'a' (for all states and actions) [44]. Figure 2 shows the RL of the production scheduling cycle.

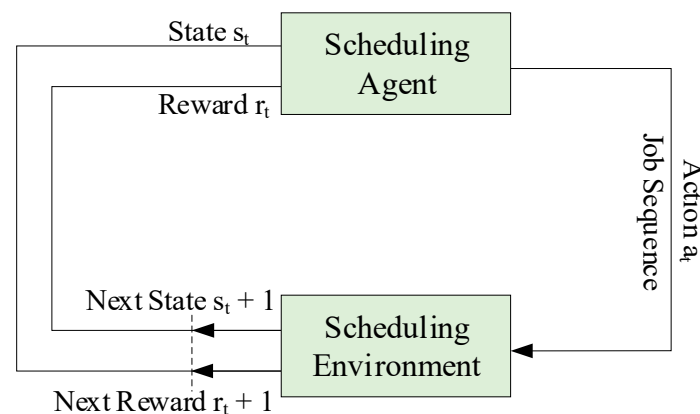


Figure 2. RL production scheduling cycle.

The learning methodologies of RL algorithms or agents can be classified into two distinct categories, model-based and model free. Model-based RL is often known as indirect learning. The agent uses a predictive model to learn the control policy from the environment through a limited number of interactions. The agent then applies this model to subsequent episodes to obtain rewards [15]. Model-free RL, or direct learning, refers to learning where an agent learns to make decisions without explicitly building a model of the environment. The agent acquires knowledge of the control policy through experiential learning from the environment, employing trial and error methods to optimize rewards without relying on any pre-existing model. This approach showcases the adaptability of RL algorithms, allowing them to learn and evolve in dynamic environments [45].

As there are many different RL algorithms, it is sensible to understand the difference among them. For this purpose, classification systems have been established to categorize them by different criteria. For instance, ALMahamid and Grolinger [46] proposed to categorize RL algorithms based on the environment type. RL algorithms can also be classified from the perspective of policy: on-policy vs. off-policy learning [47,48]. RL are mostly classified according to their learning approaches [15,45]. With this regard, updated classification of RL algorithms from the perspective of learning approaches used is provided in Figure 3.

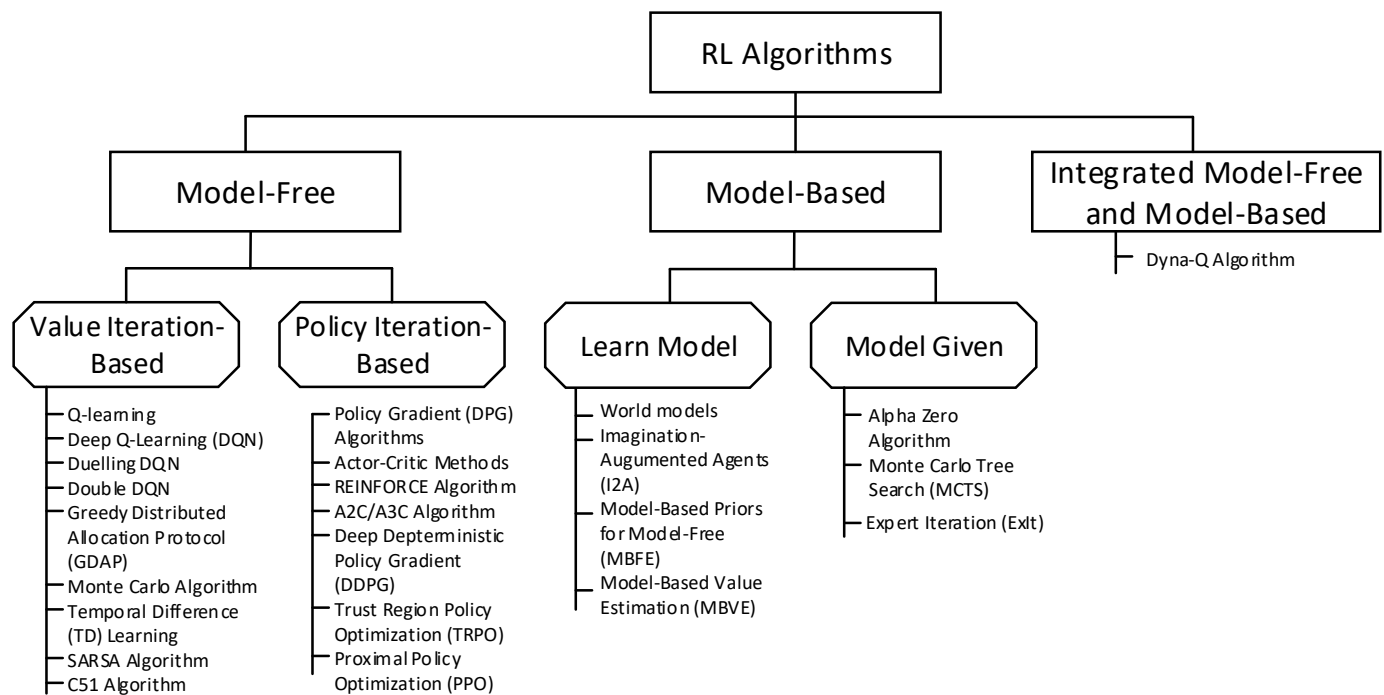


Figure 3. Classification of RL algorithms.

5. State of the Art of Reinforcement Learning in Production Scheduling

Deterministic scheduling encompasses organizing and planning tasks under fixed parameters and known constraints. Its goal is to optimize production efficiency and minimize turnaround times. A comprehensive work [49] explores fundamental principles and advanced techniques in deterministic scheduling, providing insights into various algorithms and their applications. Pinedo's work [50] thoroughly reviews deterministic scheduling theory, algorithms, and practical systems, highlighting their significance in manufacturing and service operations. Flowshop scheduling involves sequencing operations on multiple machines in a fixed order to minimize makespan or total completion time. Allahverdi [51] discusses various flowshop scheduling problems, including setups, and reviews algorithms and approaches to address them. Panwalkar and Smith [52] provide a seminal survey of classic and contemporary research on flowshop scheduling, covering both exact and heuristic methods.

Reinforcement learning (RL) techniques have gained popularity, affirming the interest in agent-based models. Previous studies have primarily focused on using RL to solve job-shop scheduling challenges. The manufacturing sector faces challenges such as customer satisfaction, system degradation, sustainability, inventory, and efficiency, impacting plant sustainability and profitability. Industry 4.0 and smart manufacturing offer solutions for optimized operations and high quality products. Paraschos et al. [53] integrate RL with lean green manufacturing to create a sustainable production environment, reducing environmental impact through minimized material consumption and lifecycle extension via pull production, predictive maintenance, and circular economy practices. Rigorous experimental analysis validates its effectiveness in enhancing sustainability and material reuse.

Recently, significant progress has been made in using RL to tackle several combinatorial optimization problems, including production scheduling, Vehicle Routing Problem, and Traveling Salesman Problem [54–59]. In RL, a production scheduling task can be viewed as an environment in which an agent operates, developing a policy through offline training by interacting with this environment. This approach offers a novel way to address scheduling challenges, requiring stringent real time constraints, such as dynamic job shop scheduling problems [22,60–64].

In production scheduling problems, value based RL algorithms are commonly used, including Q-learning, temporal difference TD(λ) algorithm, SARSA, ARL, informed Q-learning, dual Q-learning, approximate Q-learning, gradient descent TD(λ) algorithm, revenue sharing, Q-III learning, relational RL, relaxed SMART, and TD(λ)-learning. In Deep Reinforcement Learning (DRL), many value-based approaches are employed, such as DQN (Deep Q-Learning Networks), loosely-coupled DRL, multiclass DQN, and the Q-network algorithm [43,44,65–73].

Qu et al. [25] applied multi-agent approximation Q-learning, demonstrating its effectiveness through numerical experiments in static and dynamic environments and various flow shop scenarios. The goal was to create and implement an efficient manufacturing schedule considering realistic interactions between labor skills and adaptive machinery. Luo [74] uses the DRL method to address dynamic flexible job shop scheduling, focusing on scenarios to reduce tardiness. Luo et al. [75] first employed ‘hierarchical multi-agent proximal policy optimization’ (HMAPPO) for the constantly changing partial-no-wait multi-objective flexible job shop problem (MOFJSP) with new job insertions and machine breakdowns.

Machine deterioration during production, increases operational expenses and causes workflow disruptions, necessitating costly corrective maintenance. Preventive maintenance prolongs machine life but entails downtime and costs. The study [76] addresses these challenges by optimizing production and maintenance schedules in multi-machine systems, introducing knowledge enhanced Reinforcement Learning to enhance RL effectiveness in guiding production decisions and fostering machine collaboration. Comparative evaluations in deterministic and stochastic scenarios highlight the algorithm’s ability to maximize business rewards and prevent failures.

According to Wang et al. [54], the production scheduling process involves manufacturing various items using a hybrid production pattern incorporating the multi-agent Deep Reinforcement Learning (MADRL) model. Popper et al. [60] proposed MADRL to optimize flexible production plants considering factors like efficiency and environmental targets. Du et al. [77] used the Deep Q-Network (DQN) method to address the flexible task shop scheduling problem (FJSP) amid changing processing rates, setup time, idle time, and task transportation. This approach integrates state indicators and actions to enhance the DQN component’s efficiency. Additionally, it includes a problem driven exploratory data analysis (EDA) component to improve data exploration.

Li et al. [78] used a Deep Reinforcement Learning (DRL) method to solve the discrete flexible job shop problem with inter tool reusability (DFJSP-ITR), addressing the multi-objective optimization problem of minimizing combined makespan and total energy consumption. The proposed solution includes few generic state characteristics, a genetic programming based action space, and a reward function. Zhou et al. [79] proposed using online scheduling strategies based on RL with composite reward functions to improve industrial systems effectiveness and robustness. The work [18] utilized an advanced DRL algorithm to optimize production scheduling in complex job shops, highlighting benefits like enhanced adaptability, global visibility, and optimization. Some authors have focused on developing various DRL algorithms capable of formulating complex strategies for production scheduling. For instance, Luo et al. [26] created an online rescheduling framework for the dynamic multi-objective flexible job shop scheduling problem, enabling minimization of total tardiness or maximization of machine usage rate. Zhou et al. [27] used the deep Q-learning technique to address dynamic scheduling in intelligent manufacturing. Another dynamic scheduling method using deep RL is proposed in [80], employing proximal policy optimization to determine the ideal scheduling policy. Wang and Usher [81] examined using the reinforcement Q-learning algorithm for agent based production scheduling. The integration of RL with production scheduling signifies a major advancement in optimizing manufacturing processes. As research and technology continue to evolve, it promises even greater efficiency and adaptability in the industry. Delving into bibliometric data can

provide valuable insights into research trends, influential works, and key contributors in this rapidly developing field.

Industry 4.0 and Smart Industry lead contemporary manufacturing and production. Industry 4.0 integrates advanced technologies like the Internet of Things (IoT), Artificial Intelligence (AI), Big Data, and Cyber-Physical Systems to create highly automated and networked production environments. The concept of intelligent industry emphasizes real-time data analytics, predictive maintenance, and adaptable manufacturing processes to improve efficiency, flexibility, and production [27]. Implementing RL algorithms is crucial in these paradigms, facilitating machine learning and future prediction through interactions in the production environment. This enhances scheduling, resource allocation, and quality control, enabling firms to achieve high automation and precision in decision-making, leading to more resilient and responsive modern manufacturing [54]. Table 1 illustrates the contribution of RL algorithms to production scheduling

Table 1. Contributions of reinforcement learning on production scheduling.

Category: Environment		
Scheduling Type	Reward	Contribution
Complex Job Shop [18]	Penalties	Advanced DRL algorithm optimizing production scheduling, highlighting adaptability, visibility, and optimization.
Flow Shop [25]	Finished order vs. overdue penalty	Efficient manufacturing schedule considering labour skills and adaptive machinery.
Flexible Job Shop [74]	Tardiness/machine utilization rate; energy cost; late products	Online rescheduling minimizing tardiness or maximizing machine usage. MARL method optimizing for efficiency and environmental targets.
Job Shop [54]	Penalties	DQN technique for minimizing task completion time in dynamic scheduling. MADRL optimizing system-level performance considering workpiece interactions.
Category: Method		
Complex Job Shop [18]	Penalties	DRL algorithm for optimizing scheduling, improving adaptability, visibility, and optimization.
Flow Shop [25]	Finished order vs. overdue penalty	Efficient schedule considering labour skills and machinery interactions.
Flexible Job Shop [24]	Tardiness/machine utilization rate; energy cost; late products	Reinforcement learning minimizing completion times in dynamic job scheduling.
Category: Reward		
Complex Job Shop [62]	Penalties	Job scheduling with multi agent in resource pre-emption learning
Flow Shop [60]	Finished order vs. overdue penalty; energy cost; late products	Optimizing production plants for efficiency and sustainability.
Flexible Job Shop [19]	Tardiness/machine utilization rate; energy cost; late products	Addressing reinforcement algorithm in scheduling, aiming at decision making in Flexible Job Shop.

6. Bibliometric Analysis of Studies on RL in Production Scheduling

The bibliometric analysis of RL in the context of production scheduling is focusing here on its theoretical foundations and practical implications, between the years 1996 and 2024. The first two phases involve the collection of bibliographic data, which was gathered via Scopus database. The search was restricted to this database due to its prominence as a comprehensive repository of scientific literature and its frequent utilization in academic assessments. The inclusion criteria for the purpose of analysis presented in Sections 6.1–6.4 encompassed publications that contained the terms “reinforcement learning” and ‘production’ and ‘scheduling’. Besides that, the terms “reinforcement learning” and “deterministic scheduling” or “reinforcement learning” and “uncertain scheduling” were used for analysis carried out in Section 6.5. In the next two phases of the data sorting and presentation, the

Microsoft Excel and VOSviewer software 1.6.20 were employed to extract the necessary information, such as the annual scientific outputs, most relevant sources, most cited author, and keyword co-occurrence.

The process of selected studies for mentioned research methodology is synthetized in Figure 4, according to PRISMA guidelines.

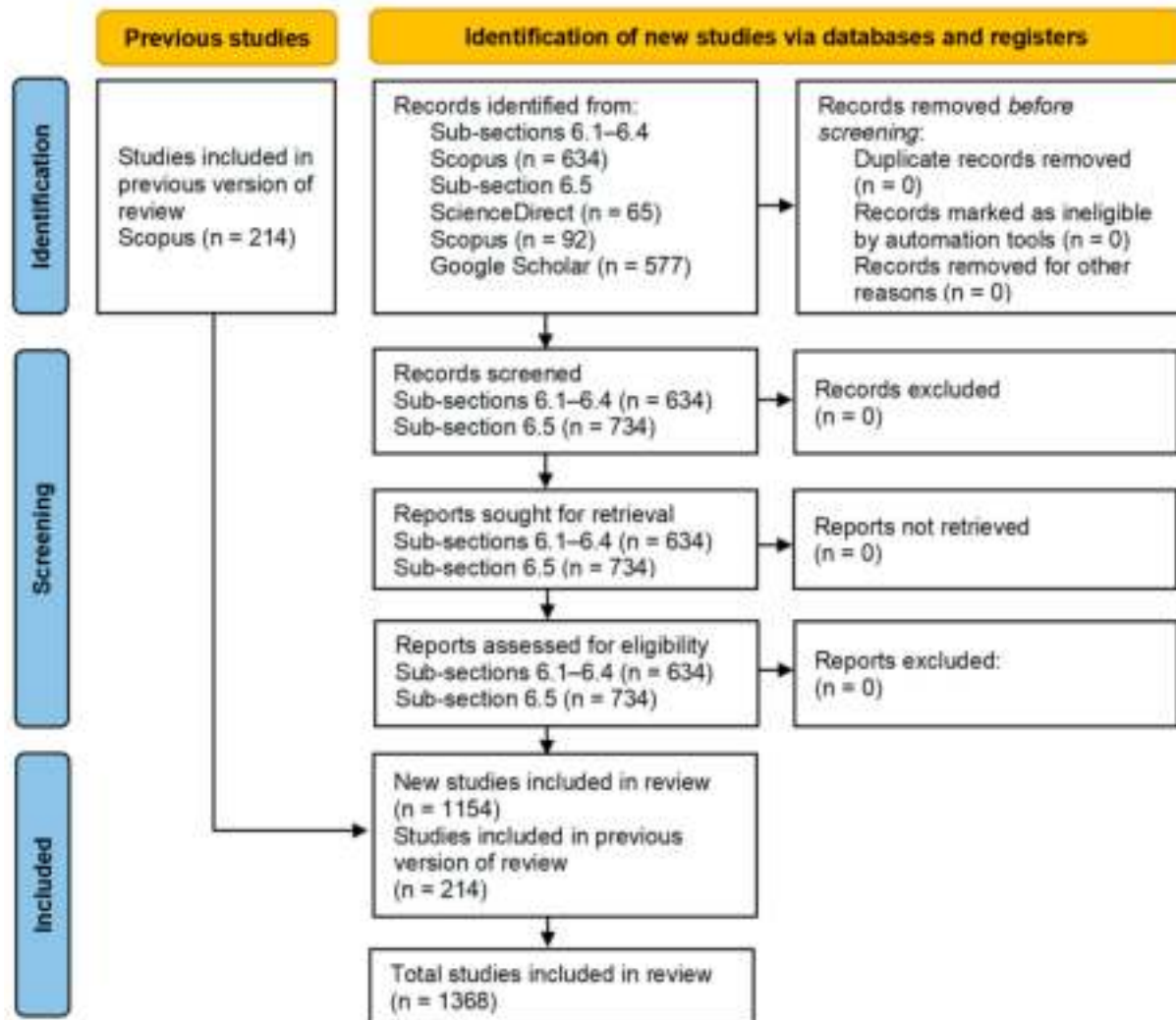


Figure 4. PRISMA 2020 flow diagram for updated systematic reviews which included searches of databases.

Using the above-specified keywords and search criteria (for Sections 6.1–6.4 $n = 634$ publications and for Section 6.5 $n = 734$ publications), together 1368 co-authored articles were found until 27 May 2024, when all document types were included in review analysis (new studies $n = 1154$ and studies included in previous version of review $n = 214$).

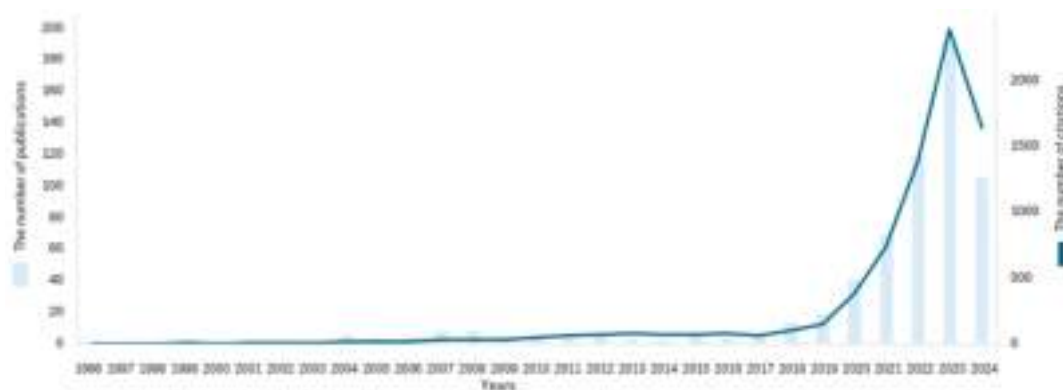
Review analysis of Sections 6.1–6.4 included 634 co-authored articles searched within “Article title, Abstract and Keywords”. The primary research areas that receive significant attention in the articles include Computer Science, Engineering, Mathematics, Decision Sciences, Business, Management and Counting, Energy, Chemical Engineering, Material Science, and other topics. The 634 publications in our sample were categorized into 19 distinct research topics. The eight primary research areas, along with their article distribution are displayed in Table 2.

Table 2. Associated research disciplines.

Research Areas	Number of Publications	Percentage
Computer Science	429	32.7%
Engineering	402	30.6%
Mathematics	159	12%
Decision Sciences	85	6.5%
Business, Management and Counting	49	3.7%
Energy	44	3.4%
Chemical Engineering	34	2.6%
Materials Science	25	2%
Others	85	6.5%

6.1. Trends of Publications and Citations

Numbers of published articles and their citations usually provide sufficiently reliable information to anticipate further development of examined research domain. Considering the numbers of publications and citations in the field of production planning using learning algorithms keeping around 29 years of data, the trend analysis graph has been derived. For this purpose, the same search terms and keywords were applied as in case of identification of major research categories, but the types of documents were extended to all the types. The reason of changing document types was to find out the initial research initiatives in this domain. The same search conditions were applied in the rest of the paper to include the larger sample of publications for the purpose of the investigation. The annual distribution of publications (out of the total 634 items) and their citations from the same database during the period from 1996 to 2024 is illustrated in Figure 5.

**Figure 5.** Publication and citation trends for RL in production scheduling field from 1996 to 2024.

An examination of the yearly scientific outputs between 1996 and 2018 amply demonstrated a relatively stable low number of articles published annually. Throughout these two decades, the need for RL in production scheduling in real conditions apparently did not appear. During the 2019–2023 era, there was a noticeable rise in the number of publications that were registered in the most recognized database for peer reviewed content. This phenomenon can be primarily attributed to the advancements in artificial intelligence. It is noteworthy to emphasize that if this exponential trend of increasing the number of publications continues as can be seen in year 2024, then one can anticipate that during the next decade the importance of RL in manufacturing scheduling will significantly increase.

6.2. Most Relevant Sources

An identification of the most relevant publications from an initial dataset presents common approach in bibliometric research since such sources usually publish influential research that attracts widespread interest. As a rule, the most productive journals have the greatest influence on the development of science in a particular field since they

publish more articles and generate more citations [77–79]. As for as the relevancy of literature in the explored field, the top ten journals that have published the most articles, are identified here. Also, the top ten most cited scientific journals are mentioned in this Sub-section. Figures 6 and 7 categorize journals according to these two criteria to show that they represent documents that exhibit the utmost relevance to RL in production scheduling.

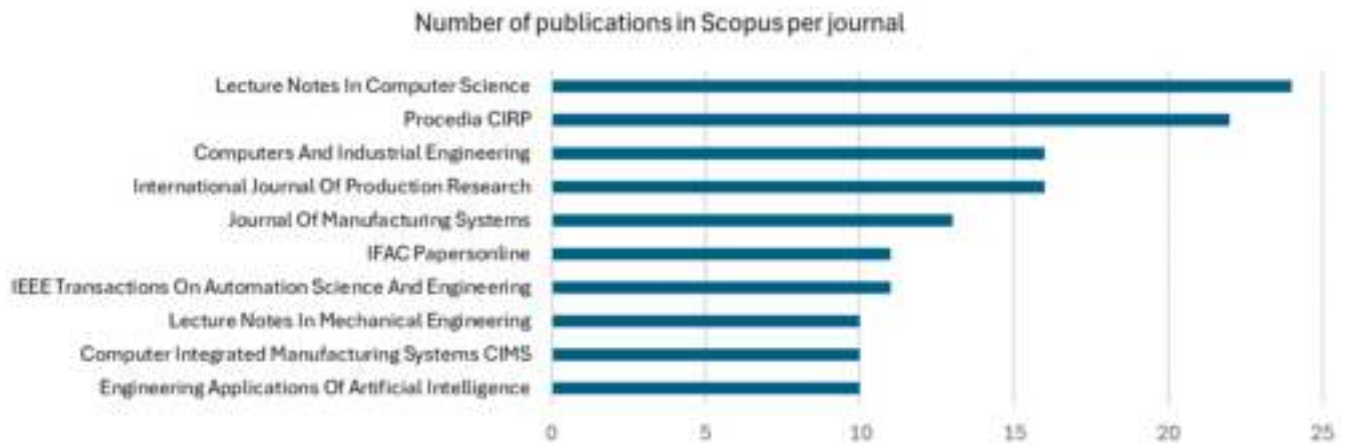


Figure 6. Top ten published journals for RL on production scheduling field from 1996 to 2024.

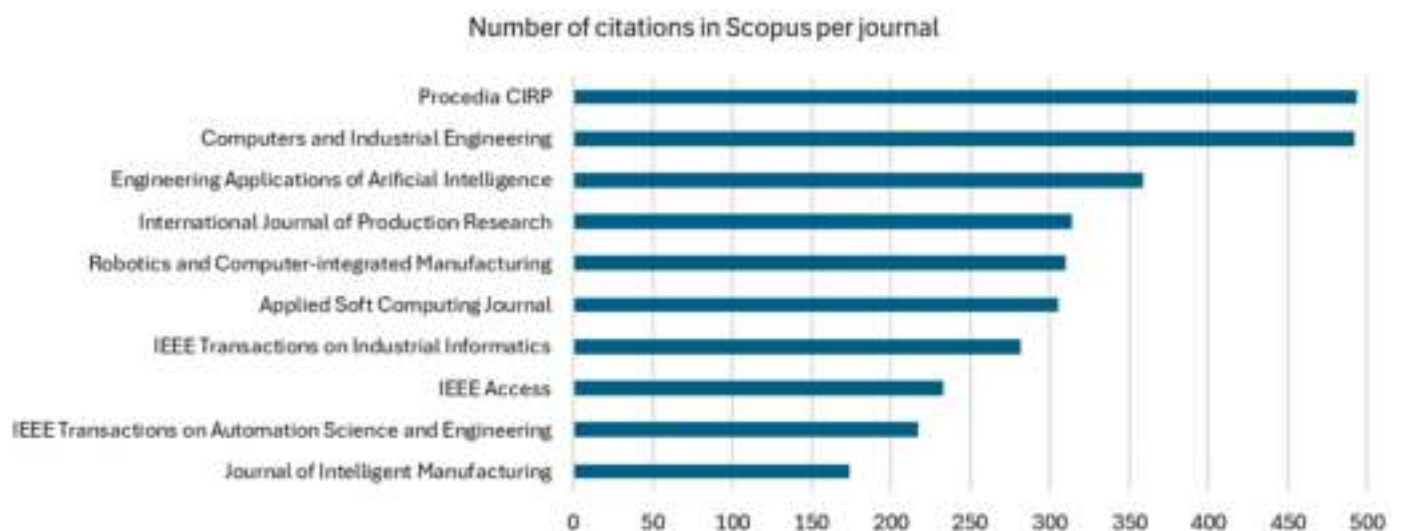


Figure 7. Top ten cited journals for RL on production scheduling field from 1996 to 2024.

The Lecture Notes in Computer Science, Procedia CIRP and Computers and Industrial Engineering along with the twelve other journals are considered the most relevant scientific publications in this field. It can also be noted that all fifteen journals listed in Figures 6 and 7, regardless of the results of the metrics used, can be empirically ranked as widely recognized for disseminating advanced research on reinforcement learning applied to production scheduling. In addition, their scientific rigor is also indicated by the fact that out of the fifteen identified journals, eight met high standards for quality as they are indexed for Current Content Connect journals with a verifiable impact on steering research practices and behaviors [80,81].

6.3. Most Cited Authors

Since the 1990s, many authors have made significant contributions to the development of this field. In this Sub-section, the intention is to present some of those authors who made significant intellectual contributions to the research. The analysis of the most cited authors

Table 3. Most influential authors.

Name	Number of Citations	Country	Name	Number of Citations	Country
Waschenck, B.	386	Germany	Wang, Y. C.	265	Taiwan
Wang, L.	375	China	Wang, H.	224	China
Wang, J.	354	China	Lin, C.C.	216	Taiwan
Bauernhansl, T.	342	Germany	Li, S.	209	China
Usher, J.M.	265	USA	Beldjilali, B.	207	Algeria

6.4. Identification of the Related Research Areas

The goal of this part of the article is to identify areas of research in which the issue of production scheduling based on RL is of interest. For this purpose, the keyword analysis application has been employed to help separate important research themes which has received a high interest of researchers from less important ones. To obtain relevant and representative categories not including less significant ones, the following setting has been used: Minimum number of occurrence of keywords—14. Based on this restriction, 88 keywords meet the threshold from the total 3767 keywords. Those main keywords that were produced automatically from the titles in the papers on production scheduling based on RL along with their occurrence are shown in Figure 9.

As can be seen, topics can be divided into five clusters based on a computer algorithm, while each cluster has a different color as shown in Figure 9. The keywords co-occurrence map highlights these clusters where the darker the color, the greater the density value is. It allows to identify relevant research topics and their mutual relationships. Based on the obtained bibliometric results extracted from VOSviewer, ten related topics that are very close to the explored research domain were identified as shown in Table 4.

The results from Table 4 point to their consistency with practical reality and operation research goals. For example, in recent years, there has been evidently increased interest in using reinforcement learning for optimization of real-time job scheduling tasks [76–86]. This fact can be correlated with the continuing trend of mass customization in the production of consumer goods [87,88]. As known, for mass customization is characteristic to meet dynamically changing user requirements in time, while customized products need to be completed by different deadlines. Accordingly, efficient real-time job-scheduling algorithms based on DRL become essential. The next important method that is ranked among the top 10 co-occurrences keywords is production control. It uses different control techniques to meet production targets regarding production schedules and quality (see, e.g., [89–92]). The next important co-occurred keyword in Table 3—multi agent systems. In general, incorporating multi-agent systems into reinforcement learning for production scheduling offers numerous advantages in terms of flexibility, scalability, and adaptability [93–95]. By enabling decentralized decision-making and continuous learning, these systems can effectively handle the complexities and dynamics of modern production environments leading to more efficient and resilient scheduling solutions [96]. Among the co-occurrences keywords it is possible to highlight also ‘smart manufacturing’ that represents the implementation domain of production scheduling based on RL. Even though smart manufacturing has also become a buzzword, which also has its drawbacks, this conception is gradually being established as the new manufacturing paradigm. On the other hand, complexity of smart manufacturing network infrastructures becomes higher and higher, and the uncertainty of such manufacturing environment becomes a serious problem [25]. These facts lead to the necessity of applying advanced dynamic planning solutions that also includes production scheduling using RL. This paragraph simultaneously answers to the main research question formulated in Section 1.

Further, application of RL in production scheduling will be here analyzed through bibliometric means from a viewpoint of different scheduling problems.

are largely adopted to solve dynamic scheduling problems. Scheduling problems can also be categorized by the nature of scheduling environment as deterministic, when processing parameters are known and invariable, and non-deterministic, when input parameters are uncertain [97]. Production scheduling problems under this second criterion can be classified as shown in Figure 10.

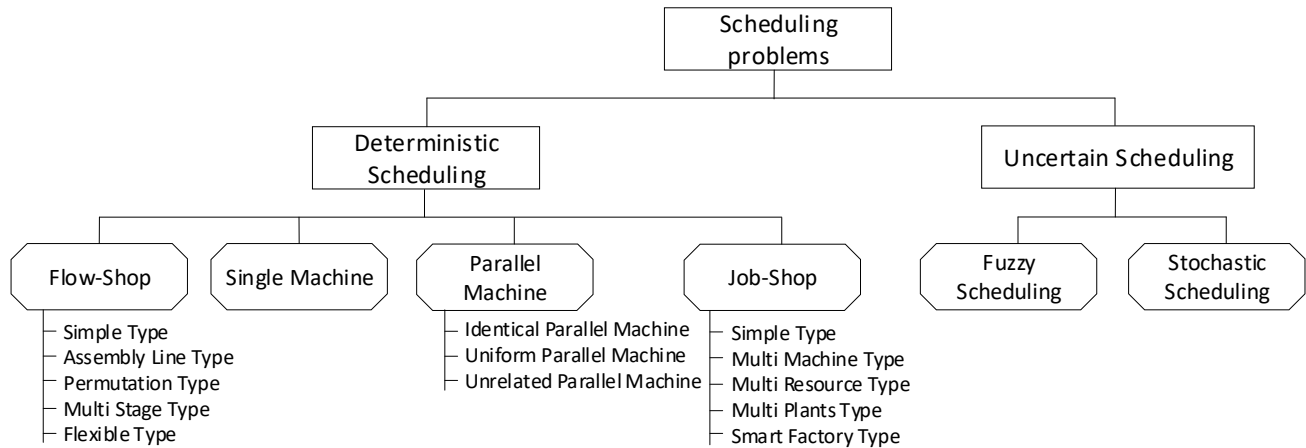


Figure 10. Basic classification of scheduling problems.

Understanding differences among specific scheduling problems is crucial for selection of the appropriate optimization and scheduling techniques with the aim to improve efficiency and productivity in various industries. The main difference among scheduling problems can be seen, e.g., in routing flexibility and complexity. For instance, job-shop problems have a high routing flexibility and complexity, while for flow-shop and single machine problems is typical low flexibility, and parallel machine problem has moderate routing flexibility and complexity [98,99]. Their differences also lie in nature of application domains for which they are intended. For example, for a customized manufacturing is typical job-shop scheduling [100,101], for assembly lines is mostly considered flow-shop scheduling problems [102,103], and open job shop scheduling is frequently applicable in healthcare [104,105].

The classification is further used to explore an application frequency of RL in the identified categories of the scheduling problems. First, deterministic versus uncertain scheduling is compared according to this view. For this purpose, the search terms were defined by combining the following keywords: “Reinforcement Learning” along with “Deterministic scheduling” or “Uncertain scheduling”, respectively. Data together with inclusion criteria were collated by searching: (i) ScienceDirect database—All fields, All years, and All document types; (ii) Scopus database—All fields, All years, All document types, and (iii) Google Scholar database—All years and All document types. Together 734 publications were analyzed. Web of Science was not used in this procedure due to the low occurrence of scientific works on the subjects. The obtained data in this way are provided in Table 5.

Table 5. Comparison of deterministic and uncertain scheduling methods based on RL.

Search Terms Used	Science Direct Database	Scopus Database	Google Scholar Database
“Deterministic scheduling” and “reinforcement learning”	53	68	474
“Uncertain scheduling” and “reinforcement learning”	12	24	103

From Table 5 is clear that deterministic scheduling based on RL is in a dominant position against uncertain scheduling using RL. Therefore, to identify the most promising

area(s) of production scheduling in the context of RL, deterministic scheduling and uncertain scheduling approaches has been examined in the following. Key words used for that purpose were:

- (a) Scheduling AND “reinforcement learning” AND specific deterministic scheduling problems (“job shop”, “flow shop”, “open shop”, “single machine”, and “parallel machines”).
- (b) “Reinforcement learning” AND specific uncertain scheduling problems (“fuzzy scheduling” and “stochastic scheduling”).

Data were retrieved from ScienceDirect database using filters ‘Research articles’, and ‘All years’; Scopus database using filter ‘Article titles’, ‘Abstracts’, ‘Keywords’, All years, and All document types, and Google Scholar database—All years and All document types. Obtained results are graphically presented and compared in percentage with each other in Figure 11.

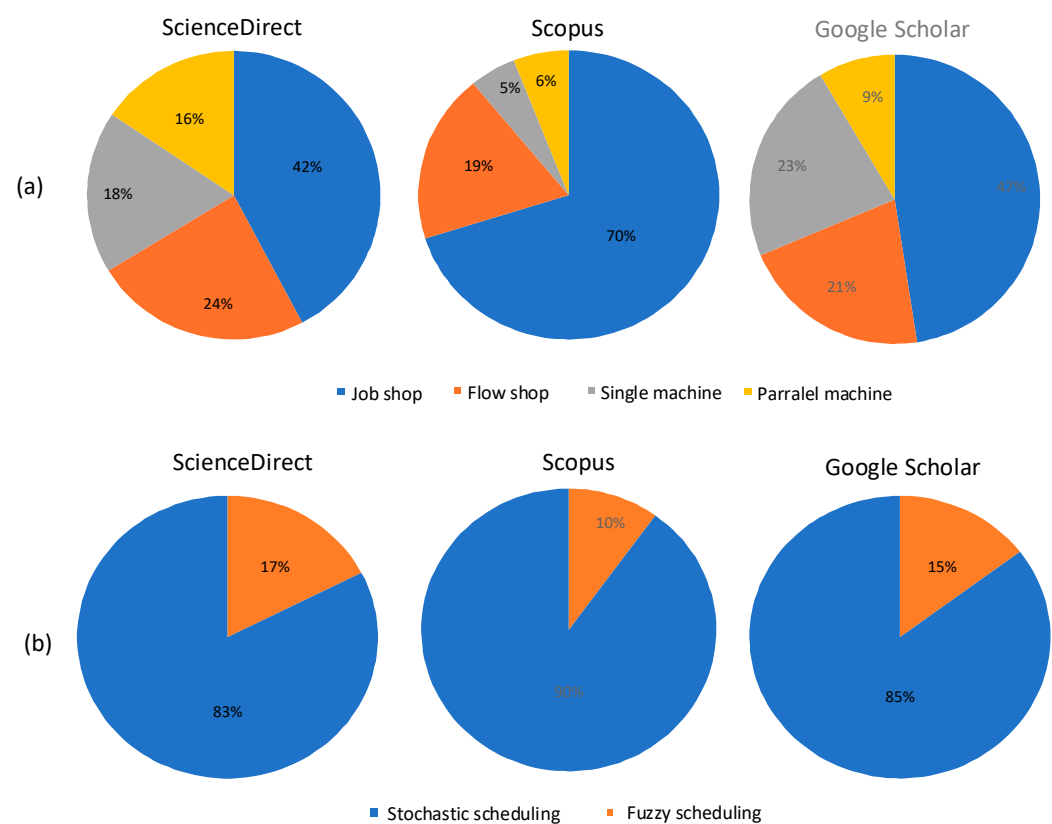


Figure 11. Comparison of (a) deterministic scheduling methods and (b) uncertain scheduling methods from their occurrences in literature.

One can see from Figure 11a that RL-based scheduling algorithms are mostly used to solve job shop scheduling problems [106], while flow shop production environment is the second most important type of scheduling problems, where RL is applied. Obviously, RL-based algorithms found application in other types of scheduling problems. As the most noticeable of them, it is possible to mention energy efficiency scheduling, multi-objective scheduling, and distributive scheduling. The last-mentioned scheduling problem is exploited especially in intelligent manufacturing systems [107]. Analyzing the results from Figure 11b, it is observed that RL is predominantly used in stochastic scheduling methods comparing with its application in fuzzy scheduling methods. Moreover, the effectiveness of stochastic scheduling using RL has been also demonstrated through benchmarking studies [29,108,109]. Although the application of RL in fuzzy scheduling is promising, it remains less common due to additional complexities involved in integrating fuzzy logic

with RL methods [110,111]. However, the results of this quantitative approach for comparison of application of RL in different scheduling problems provide some useful managerial insights into scheduling practice. Nevertheless, further exploration and research on qualitative aspects of RL-based scheduling, particularly in the context of smart manufacturing, is required.

7. Comparison of the Presented Research and Previous Review Articles

As was already mentioned in introduction, this review paper continues the work of the two earlier articles [28,29] by analyzing some new relevant aspects of production scheduling methods using RL and provides some updated information regarding recent developments in the field. The purpose of this section is to provide a clear view on differences and similarities of the three works in terms of their subject matters. The following Table 6 contains a summary comparison of investigation areas included in the mentioned works.

Table 6. The comparison of the review papers on RL in Production Scheduling.

The Focus of Investigation	The Compared Review Papers		
	Wang et al., 2021 [28]	Kayhan and Yildiz, 2021 [29]	Offered Research
The citation trend for RL in production scheduling from 1996 to 2024			✓
The most relevant publications in the field	✓		✓
The most influential authors in the field			✓
The comparison of deterministic and uncertain scheduling methods based on RL			✓
The comparison of stochastic and fuzzy scheduling methods based on RL			✓
The publication trend for RL in production scheduling	✓	✓	✓
The most relevant sources published research on RL in production scheduling			✓
The comparison of the most frequent deterministic scheduling methods based on RL	✓	✓	✓
The comparison of the value-based and policy-based scheduling methods based on RL	✓		
The comparison of static and dynamic scheduling methods based on RL	✓		
The categorization of the literature on RL in production scheduling according to multiple aspects		✓	
The publication trend for RL in production scheduling according to manufacturing environment		✓	
The comparison of different types of RL algorithms for production scheduling		✓	
The comparison of stochastic and deterministic scheduling methods based on RL		✓	

From this table it can be seen that the compared papers are mostly complementary to each other.

8. Conclusions

The existing body of research on production scheduling primarily consists of studies conducted within the domains of Computer Science, Engineering, Mathematics, Decision Sciences, Business, Management and Counting, Energy, Chemical Engineering, and Material Science. These scientific disciplines, notably ‘Engineering’ and ‘Computer Science’ have exerted significant influence on development of production scheduling based on RL.

The examination of bibliometric findings frequently indicates that an increase in the quantity of published articles is associated with recognition of progressive trends in the subject. Also, for this reason, a bibliometric analysis is becoming more and more beneficial in a variety of academic fields since it makes mapping scientific information and analyzing research development objective and repeatable. The use of this method enables us to identify the networks of scientific collaboration, to establish connections between novel study themes and research streams, as well as show the connections between citations, co-citations and published productivity in the field.

The main contribution of this article can be summarized by two steps:

- (i) This review brings additional insights into RL in production scheduling by providing the following new features:
 - The citation trend for RL on production scheduling field from 1996 to 2024 (see Figure 5) has been carried out that shows that increasing trend starting from 2019 to present rapidly continues in noticeable rise in their number.
 - The analysis of the most relevant sources from the viewpoint of number of their citations (see Figure 7) has been performed to identify the most impacted sources where the latest knowledge in this field is available.
 - The most influential authors in this domain were identified (see Table 3) for the determination of the cutting-edge state in RL-based production scheduling.
 - The quantitative comparison of deterministic and uncertain scheduling methods based on RL has been conducted (see in Table 5). It showed that deterministic scheduling methods using RL is in a dominant position against RL-based uncertain scheduling techniques.
 - The quantitative comparison of stochastic and fuzzy scheduling methods has been given in Figure 11b. From the comparison, it has been observed that RL is predominantly used in stochastic scheduling methods comparing with its application in fuzzy scheduling methods.
- (ii) The contribution of this paper can be also seen in the following activities and resulting statements:
 - The publication trend for RL in production scheduling field has been updated by mapping period from 1996 to 2024 (see Figure 5). It showed that exponential growth of the publications starting from 2019 [28] rapidly continues in noticeable rise in their number.
 - The updated list of the most relevant sources published research on RL in production scheduling was compiled in Figure 6. It has been found that five of ten sources identified until 2021 belong among top ten sources identified until 2024. It means that due to the growing interest about this research field, at the same time, the number of journals covering this topic is increasing.
 - Actualization of the quantitative comparison of the most frequent deterministic scheduling methods based on RL was provided in Figure 11a. In this context, the previously identified trend has been confirmed that RL-based scheduling algorithms are mostly used to solve job shop scheduling problems, while flow shop production presents the second most important type of scheduling problems.
 - The updated categorization of newly developed RL algorithms has been elaborated and presented in Figure 3 that can provide better implementation support for decision making in real-world problems.

- The updated classification of scheduling methods is provided in Figure 10. In principle, it can be used for easier selection of the scheduling techniques to solve specific types of problems.

In addition to the above-mentioned findings, it would be needed to focus on other challenges to be considered in the future such manufacturing process planning with integrated support for knowledge sharing, increasing demand for improvements in ubiquitous “smartness” in manufacturing processes including designing and implementing smart algorithms, and the need for robust scheduling tools for agile collaborative manufacturing systems.

Author Contributions: Conceptualization, V.M.; methodology, R.S. and A.B.; validation, Z.S.; formal analysis, V.M. and R.S.; investigation, V.M. and A.B.; writing—original draft preparation, A.B. and V.M.; writing—review and editing, R.S. and V.M.; visualization, Z.S.; project administration, V.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research has been funded by the project SME 5.0 with funding received from the European Union’s Horizon research and innovation program under the Marie Skłodowska-Curie Grant Agreement No. 101086487 and the KEPA (Cultural and Education Grant Agency) project No. 044TUKE-4/2023 granted by the Ministry of Education of the Slovak Republic.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data is contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Pinedo, M. *Planning and Scheduling in Manufacturing and Services*; Springer: New York, NY, USA, 2005.
2. Beheshti, Z.; Shamsuddin, S.M.H. A review of population-based meta-heuristic algorithms. *Int. J. Adv. Soft Comput. Appl.* **2013**, *5*, 1–35.
3. Khafa, F.; Abraham, A. (Eds.) *Metaheuristics for Scheduling in Industrial and Manufacturing Applications*; Springer: Berlin/Heidelberg, Germany, 2008; Volume 128.
4. Abdel-Kader, R.F. Particle swarm optimization for constrained instruction scheduling. *VLSI Des.* **2008**, *2008*, 930610. [[CrossRef](#)]
5. Balamurugan, A.; Ranjitharamasamy, S.P. A Modified Heuristics for the Batch Size Optimization with Combined Time in a Mass-Customized Manufacturing System. *Int. J. Ind. Eng. Theory Appl. Pract.* **2023**, *30*, 1090–1115.
6. Ghassemi Tari, F.; Olfat, L. Heuristic rules for tardiness problem in flow shop with intermediate due dates. *Int. J. Adv. Manuf. Technol.* **2014**, *71*, 381–393. [[CrossRef](#)]
7. Modrak, V.; Pandian, R.S. Flow shop scheduling algorithm to minimize completion time for n-jobs m-machines problem. *Teh. Vjesn.* **2010**, *17*, 273–278.
8. Thenarasu, M.; Rameshkumar, K.; Rousseau, J.; Anbuudayasankar, S.P. Development and analysis of priority decision rules using MCDM approach for a flexible job shop scheduling: A simulation study. *Simul. Model. Pract. Theory* **2022**, *114*, 102416. [[CrossRef](#)]
9. Pandian, S.; Modrak, V. Possibilities, obstacles and challenges of genetic algorithm in manufacturing cell formation. *Adv. Logist. Syst.* **2009**, *3*, 63–70.
10. Abdulredha, M.N.; Bara’a, A.A.; Jabir, A.J. Heuristic and meta-heuristic optimization models for task scheduling in cloud-fog systems: A review. *Iraqi J. Electr. Electron. Eng.* **2020**, *16*, 103–112. [[CrossRef](#)]
11. Modrak, V.; Pandian, R.S.; Semanco, P. Calibration of GA parameters for layout design optimization problems using design of experiments. *Appl. Sci.* **2021**, *11*, 6940. [[CrossRef](#)]
12. Keshanchi, B.; Souri, A.; Navimipour, N.J. An improved genetic algorithm for task scheduling in the cloud environments using the priority queues: Formal verification, simulation, and statistical testing. *J. Syst. Softw.* **2017**, *124*, 1–21. [[CrossRef](#)]
13. Jans, R.; Degraeve, Z. Meta-heuristics for dynamic lot sizing: A review and comparison of solution approaches. *Eur. J. Oper. Res.* **2007**, *177*, 1855–1875. [[CrossRef](#)]
14. Han, B.A.; Yang, J.J. A deep reinforcement learning based solution for flexible job shop scheduling problem. *Int. J. Simul. Model.* **2021**, *20*, 375–386. [[CrossRef](#)]
15. Shyalika, C.; Silva, T.; Karunananda, A. Reinforcement Learning in Dynamic Task Scheduling: A Review. *SN Comput. Sci.* **2020**, *1*, 306. [[CrossRef](#)]
16. Wang, X.; Zhang, L.; Ren, L.; Xie, K.; Wang, K.; Ye, F.; Chen, Z. Brief Review on Applying Reinforcement Learning to Job Shop Scheduling Problems. *J. Syst. Simul.* **2022**, *33*, 2782–2791.

17. Dima, I.C.; Gabrara, J.; Modrak, V.; Piotr, P.; Popescu, C. Using the expert systems in the operational management of production. In Proceedings of the 11th WSEAS International Conference on Mathematics and Computers in Business and Economics (MCBE'10), Iasi, Romania, 13–15 June 2010; WSEAS Press: Stevens Point, WI, USA, 2010.
18. Waschneck, B.; Reichstaller, A.; Belzner, L.; Altenmüller, T.; Bauernhansl, T.; Knapp, A.; Kyek, A. Optimization of global production scheduling with deep reinforcement learning. *Procedia CIRP* **2018**, *72*, 1264–1269. [\[CrossRef\]](#)
19. Yan, J.; Liu, Z.; Zhang, T.; Zhang, Y. Autonomous decision-making method of transportation process for flexible job shop scheduling problem based on reinforcement learning. In Proceedings of the 2021 International Conference on Machine Learning and Intelligent Systems Engineering, MLISE, Chongqing, China, 9–11 July 2021; Institute of Electrical and Electronics Engineers Inc.: Piscataway, NJ, USA; pp. 234–238.
20. Modrak, V.; Pandian, R.S. *Operations Management Research and Cellular Manufacturing Systems*; IGI Global: Hershey, PA, USA, 2010.
21. Huang, Z.; Liu, Q.; Zhu, F. Hierarchical reinforcement learning with adaptive scheduling for robot control. *Eng. Appl. Artif. Intell.* **2023**, *126*, 107130. [\[CrossRef\]](#)
22. Arviv, K.; Stern, H.; Edan, Y. Collaborative reinforcement learning for a two-robot job transfer flow-shop scheduling problem. *Int. J. Prod. Res.* **2016**, *54*, 1196–1209. [\[CrossRef\]](#)
23. Wen, X.; Zhang, X.; Xing, H.; Ye, G.; Li, H.; Zhang, Y.; Wang, H. An improved genetic algorithm based on reinforcement learning for aircraft assembly scheduling problem. *Comput. Ind. Eng.* **2024**, *193*, 110263. [\[CrossRef\]](#)
24. Aydin, M.E.; Öztemel, E. Dynamic job-shop scheduling using reinforcement learning agents. *Robot. Auton. Syst.* **2000**, *33*, 169–178. [\[CrossRef\]](#)
25. Qu, S.; Wang, J.; Govil, S.; Leckie, J.O. Optimized Adaptive Scheduling of a Manufacturing Process System with Multi-skill Workforce and Multiple Machine Types: An Ontology-based, Multi-agent Reinforcement Learning Approach. *Procedia CIRP* **2016**, *57*, 55–60. [\[CrossRef\]](#)
26. Luo, S.; Zhang, L.; Fan, Y. Dynamic multi-objective scheduling for flexible job shop by deep reinforcement learning. *Comput. Ind. Eng.* **2021**, *159*, 107489. [\[CrossRef\]](#)
27. Zhou, L.; Zhang, L.; Horn, B.K.P. Deep reinforcement learning-based dynamic scheduling in smart manufacturing. *Procedia CIRP* **2020**, *93*, 383–388. [\[CrossRef\]](#)
28. Wang, L.; Pan, Z.; Wang, J. A Review of Reinforcement Learning Based Intelligent Optimization for Manufacturing Scheduling. *Complex Syst. Model. Simul.* **2021**, *1*, 257–270. [\[CrossRef\]](#)
29. Kayhan, B.M.; Yildiz, G. Reinforcement learning applications to machine scheduling problems: A comprehensive literature review. *J. Intell. Manuf.* **2021**, *34*, 905–929. [\[CrossRef\]](#)
30. Broadus, R.N. Toward a Definition of “Bibliometrics”. *Scientometrics* **1987**, *12*, 373–379. [\[CrossRef\]](#)
31. Arunmozhi, B.; Sudhakarapandian, R.; Sultan Batcha, Y.; Rajay Vedaraj, I.S. An inferential analysis of stainless steel in additive manufacturing using bibliometric indicators. *Mater Today Proc.* **2023**, *in press*. [\[CrossRef\]](#)
32. Randhawa, K.; Wilden, R.; Hohberger, J. A bibliometric review of open innovation: Setting a research agenda. *J. Prod. Innov. Manag.* **2016**, *33*, 750–772. [\[CrossRef\]](#)
33. van Raan, A. Advanced bibliometric methods as quantitative core of peer review based evaluation and foresight exercises. *Scientometrics* **1996**, *36*, 397–420. [\[CrossRef\]](#)
34. Brandom, R.B. *Articulating Reasons: An Introduction to Inferentialism*; Harvard University Press: Cambridge, MA, USA, 2001.
35. Kothari, C.R. *Research Methodology: Methods and Techniques*; New Age International: New Delhi, India, 2004.
36. Suárez, M. An inferential conception of scientific representation. *Philos. Sci.* **2004**, *71*, 767–779. [\[CrossRef\]](#)
37. Contessa, G. Scientific representation, interpretation, and surrogate reasoning. *Philos. Sci.* **2007**, *74*, 48–68. [\[CrossRef\]](#)
38. Govier, T. *Problems in Argument Analysis and Evaluation*; University of Windsor: Windsor, ON, Canada, 2018; Volume 6.
39. Munusamy, R.; Mukherjee, A.; Vasudevan, K.; Venkateswaran, B. Design and Simulation of an Artificial intelligence (AI) Brain for a 2D Vehicle Navigation System. *INCAS Bull.* **2022**, *14*, 53–64. [\[CrossRef\]](#)
40. Dunjko, V.; Briegel, H.J. Machine learning & artificial intelligence in the quantum domain: A review of recent progress. *Rep. Prog. Phys.* **2018**, *81*, 074001.
41. Horvitz, E.; Mulligan, D. Data, privacy, and the greater good. *Science* **2015**, *349*, 253–255. [\[CrossRef\]](#)
42. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT: New York, NY, USA, 2018.
43. Kuhnle, A.; May, M.C.; Schäfer, L.; Lanza, G. Explainable reinforcement learning in production control of job shop manufacturing system. *Int. J. Prod. Res.* **2022**, *60*, 5812–5834. [\[CrossRef\]](#)
44. Estes, A.; Peidro, D.; Mula, J.; Díaz-Madroñero, M. Reinforcement learning applied to production planning and control. *Int. J. Prod. Res.* **2023**, *61*, 5772–5789. [\[CrossRef\]](#)
45. Khan, M.A.M.; Khan, M.R.J.; Tooshil, A.; Sikder, N.; Mahmud, M.P.; Kouzani, A.Z.; Nahid, A.A. A systematic review on reinforcement learning-based robotics within the last decade. *IEEE Access* **2020**, *8*, 176598–176623. [\[CrossRef\]](#)
46. AlMahamid, F.; Grolinger, K. Reinforcement learning algorithms: An overview and classification. In Proceedings of the 2021 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE), Kingston, ON, Canada, 12–17 September 2021; IEEE: Piscataway, NJ, USA; pp. 1–7.
47. Akalin, N.; Loutfi, A. Reinforcement learning approaches in social robotics. *Sensors* **2021**, *21*, 1292. [\[CrossRef\]](#)
48. Zhang, H.; Yu, T. Taxonomy of reinforcement learning algorithms. In *Deep Reinforcement Learning: Fundamentals, Research and Applications*; Springer: Singapore, 2020; pp. 125–133.

49. Baker, K.R.; Trietsch, D. *Principles of Sequencing and Scheduling*; John Wiley & Sons: Hoboken, NJ, USA, 2009.
50. Pinedo, M.L. *Scheduling: Theory, Algorithms, and Systems*; Springer: Berlin/Heidelberg, Germany, 2016.
51. Allahverdi, A.; Ng, C.T.; Cheng, T.C.E.; Kovalyov, M.Y. A survey of scheduling problems with setup times or costs. *Eur. J. Oper. Res.* **2008**, *187*, 985–1032. [\[CrossRef\]](#)
52. Panwalkar, S.S.; Smith, M.L. Survey of flow shop scheduling research. *Oper. Res.* **1977**, *25*, 45–84. [\[CrossRef\]](#)
53. Paraschos, P.D.; Koulinas, G.K.; Koulouriotis, D.E. Reinforcement Learning-Based Optimization for Sustainable and Lean Production within the Context of Industry 4.0. *Algorithms* **2024**, *17*, 98. [\[CrossRef\]](#)
54. Wang, S.; Li, J.; Luo, Y. Smart Scheduling for Flexible and Hybrid Production with Multi-Agent Deep Reinforcement Learning. In Proceedings of the 2021 IEEE 2nd International Conference on Information Technology, Big Data and Artificial Intelligence, ICIBA, Chongqing, China, 17–19 December 2021; Institute of Electrical and Electronics Engineers Inc.: Piscataway, NJ, USA; pp. 288–294.
55. Tang, J.; Haddad, Y.; Salonitis, K. Reconfigurable manufacturing system scheduling: A deep reinforcement learning approach. *Procedia CIRP* **2022**, *107*, 1198–1203. [\[CrossRef\]](#)
56. Shahrabi, J.; Adibi, M.A.; Mahootchi, M. A reinforcement learning approach to parameter estimation in dynamic job shop scheduling. *Comput Ind Eng.* **2017**, *110*, 75–82. [\[CrossRef\]](#)
57. Yang, J.; You, X.; Wu, G.; Hassan, M.M.; Almogren, A.; Guna, J. Application of reinforcement learning in UAV cluster task scheduling. *Future Gener. Comput. Syst.* **2019**, *95*, 140–148. [\[CrossRef\]](#)
58. Yuan, X.; Pan, Y.; Yang, J.; Wang, W.; Huang, Z. Study on the application of reinforcement learning in the operation optimization of HVAC system. In *Building Simulation*; Tsinghua University Press: Beijing, China, 2021; Volume 14, pp. 75–87.
59. Kurinov, I.; Orzechowski, G.; Hämäläinen, P.; Mikkola, A. Automated excavator based on reinforcement learning and multibody system dynamics. *IEEE Access* **2020**, *8*, 213998–214006. [\[CrossRef\]](#)
60. Popper, J.; Motsch, W.; David, A.; Petzsche, T.; Ruskowski, M. Utilizing multi-agent deep reinforcement learning for flexible job shop scheduling under sustainable viewpoints. In Proceedings of the International Conference on Electrical, Computer, Communications and Mechatronics Engineering 2021, ICECCME, Mauritius, Mauritius, 7–8 October 2021; Institute of Electrical and Electronics Engineers Inc.: Piscataway, NJ, USA.
61. Xiong, H.; Fan, H.; Jiang, G.; Li, G. A simulation-based study of dispatching rules in a dynamic job shop scheduling problem with batch release and extended technical precedence constraints. *Eur. J. Oper. Res.* **2017**, *257*, 13–24. [\[CrossRef\]](#)
62. Palacio, J.C.; Jiménez, Y.M.; Schietgat, L.; Van Doninck, B.; Nowé, A. A Q-Learning algorithm for flexible job shop scheduling in a real-world manufacturing scenario. *Procedia CIRP* **2022**, *106*, 227–232. [\[CrossRef\]](#)
63. Chang, J.; Yu, D.; Hu, Y.; He, W.; Yu, H. Deep Reinforcement Learning for Dynamic Flexible Job Shop Scheduling with Random Job Arrival. *Processes* **2022**, *10*, 760. [\[CrossRef\]](#)
64. Liu, R.; Piplani, R.; Toro, C. Deep reinforcement learning for dynamic scheduling of a flexible job shop. *Int. J. Prod. Res.* **2022**, *60*, 4049–4069. [\[CrossRef\]](#)
65. Samsonov, V.; Kemmerling, M.; Paegert, M.; Lütticke, D.; Sauermann, F.; Gützlaß, A.; Schuh, G.; Meisen, T. Manufacturing control in job shop environments with reinforcement learning. In Proceedings of the 13th International Conference on Agents and Artificial Intelligence (ICAART 2021), Online, 4–6 February 2021; SciTePress: Setúbal, Portugal; pp. 589–597.
66. Cunha, B.; Madureira, A.M.; Fonseca, B.; Coelho, D. Deep reinforcement learning as a job shop scheduling solver: A literature review. In Proceedings of the 18th International Conference on Hybrid Intelligent Systems (HIS 2018), Porto, Portugal, 13–15 December 2018; Madureira, A.M., Abraham, A., Gandhi, N., Varela, M.L., Eds.; Springer International Publishing: Cham, Switzerland, 2018.
67. Wang, X.; Zhang, L.; Lin, T.; Zhao, C.; Wang, K.; Chen, Z. Solving job scheduling problems in a resource preemption environment with multi-agent reinforcement learning. *Robot. Comput. Integr. Manuf.* **2022**, *77*, 102324. [\[CrossRef\]](#)
68. Oh, S.H.; Cho, Y.I.; Woo, J.H. Distributional reinforcement learning with the independent learners for flexible job shop scheduling problem with high variability. *J. Comput. Des. Eng.* **2022**, *9*, 1157–1174. [\[CrossRef\]](#)
69. Zhang, Y.; Zhu, H.; Tang, D.; Zhou, T.; Gui, Y. Dynamic job shop scheduling based on deep reinforcement learning for multi-agent manufacturing systems. *Robot. Comput. Integr. Manuf.* **2022**, *78*, 102412. [\[CrossRef\]](#)
70. Liang, Y.; Sun, Z.; Song, T.; Chou, Q.; Fan, W.; Fan, J.; Rui, Y.; Zhou, Q.; Bai, J.; Yang, C.; et al. Lenovo Schedules Laptop Manufacturing Using Deep Reinforcement Learning. *Interfaces* **2022**, *52*, 56–68. [\[CrossRef\]](#)
71. Chen, Y.; Guo, W.; Liu, J.; Shen, S.; Lin, J.; Cui, D. A multi-setpoint cooling control approach for air-cooled data centers using the deep Q-network algorithm. *Meas. Control* **2024**, *57*, 782–793. [\[CrossRef\]](#)
72. Théate, T.; Ernst, D. An application of deep reinforcement learning to algorithmic trading. *Expert Syst. Appl.* **2021**, *173*, 114632. [\[CrossRef\]](#)
73. Sanaye, S.; Sarrafi, A. A novel energy management method based on Deep Q Network algorithm for low operating cost of an integrated hybrid system. *Energy Rep.* **2021**, *7*, 2647–2663. [\[CrossRef\]](#)
74. Luo, S. Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning. *Appl. Soft Comput. J.* **2020**, *91*, 106208. [\[CrossRef\]](#)
75. Luo, S.; Zhang, L.; Fan, Y. Real-Time Scheduling for Dynamic Partial-No-Wait Multiobjective Flexible Job Shop by Deep Reinforcement Learning. *IEEE Trans. Autom. Sci. Eng.* **2022**, *19*, 3020–3038. [\[CrossRef\]](#)

76. Hu, J.; Wang, H.; Tang, H.-K.; Kanazawa, T.; Gupta, C.; Farahat, A. Knowledge-enhanced reinforcement learning for multi-machine integrated production and maintenance scheduling. *Comput. Ind. Eng.* **2023**, *185*, 109631. [\[CrossRef\]](#)
77. Du, Y.; Li, J.Q.; Chen, X.L.; Duan, P.Y.; Pan, Q.K. Knowledge-Based Reinforcement Learning and Estimation of Distribution Algorithm for Flexible Job Shop Scheduling Problem. *IEEE Trans. Emerg. Top. Comput. Intell.* **2023**, *7*, 1036–1050. [\[CrossRef\]](#)
78. Li, Y.; Gu, W.; Yuan, M.; Tang, Y. Real-time data-driven dynamic scheduling for flexible job shop with insufficient transportation resources using hybrid deep Q network. *Robot. Comput. Integr. Manuf.* **2022**, *74*, 102283. [\[CrossRef\]](#)
79. Zhou, T.; Zhu, H.; Tang, D.; Liu, C.; Cai, Q.; Shi, W.; Gui, Y. Reinforcement learning for online optimization of job-shop scheduling in a smart manufacturing factory. *Adv. Mech. Eng.* **2022**, *14*, 16878132221086120. [\[CrossRef\]](#)
80. Wang, L.; Hu, X.; Wang, Y.; Xu, S.; Ma, S.; Yang, K.; Wang, W. Dynamic job-shop scheduling in smart manufacturing using deep reinforcement learning. *Comput. Netw.* **2021**, *190*, 107969. [\[CrossRef\]](#)
81. Wang, Y.; Usher, J.M. Application of reinforcement learning for agent-based production scheduling. *Eng. Appl. Artif. Intell.* **2005**, *18*, 73–82. [\[CrossRef\]](#)
82. Cancino, C.A.; Merigó, J.M.; Coronado, F.C. A bibliometric analysis of leading universities in innovation research. *J. Innov. Knowl.* **2017**, *2*, 106–124. [\[CrossRef\]](#)
83. Varin, C.; Cattelan, M.; Firth, D. Statistical modelling of citation exchange between statistics journals. *J. R. Stat. Soc. Ser. A Stat. Soc.* **2016**, *179*, 1–63. [\[CrossRef\]](#) [\[PubMed\]](#)
84. Moral-Muñoz, J.A.; Herrera-Viedma, E.; Santisteban-Espejo, A.; Cobo, M.J. Software tools for conducting bibliometric analysis in science: An up-to-date review. *Prof. De La Inf./Inf. Prof.* **2020**, *29*, e290103. [\[CrossRef\]](#)
85. Curry, S. Let's move beyond the rhetoric: It's time to change how we judge research. *Nature* **2018**, *554*, 147–148. [\[CrossRef\]](#) [\[PubMed\]](#)
86. Al-Hoorie, A.; Vitta, J.P. The seven sins of L2 research: A review of 30 journals' statistical quality and their CiteScore, SJR, SNIP, JCR Impact Factors. *Lang. Teach. Res.* **2019**, *23*, 727–744. [\[CrossRef\]](#)
87. Waltman, L.; Van Eck, N.J.; Noyons, E.C.M. A Unified Approach to Mapping and Clustering of Bibliometric Networks. *J. Informetr.* **2010**, *4*, 629–635. [\[CrossRef\]](#)
88. Cheng, F.; Huang, Y.; Tanpure, B.; Sawalani, P.; Cheng, L.; Liu, C. Cost-aware job scheduling for cloud instances using deep reinforcement learning. *Clust. Comput.* **2022**, *25*, 619–631. [\[CrossRef\]](#)
89. Thaipisutikul, T.; Chen, Y.-C.; Hui, L.; Chen, S.-C.; Mongkolwat, P.; Shih, T.K. The matter of deep reinforcement learning towards practical AI applications. In Proceedings of the 12th International Conference on Ubi-Media Computing, Bali, Indonesia, 5–8 August 2019; pp. 24–29.
90. Yan, J.; Huang, Y.; Gupta, A.; Gupta, A.; Liu, C.; Li, J.; Cheng, L. Energy-aware systems for real-time job scheduling in cloud data centers: A deep reinforcement learning approach. *Comput. Electr. Eng.* **2022**, *99*, 107688. [\[CrossRef\]](#)
91. Piller, F.T. Mass customization: Reflections on the state of the concept. *Int. J. Flex. Manuf. Syst.* **2004**, *16*, 313–334. [\[CrossRef\]](#)
92. Suzić, N.; Forza, C.; Trentin, A.; Anišić, Z. Implementation guidelines for mass customization: Current characteristics and suggestions for improvement. *Prod. Plan. Control* **2018**, *29*, 856–871. [\[CrossRef\]](#)
93. Altenmüller, T.; Stüker, T.; Waschneck, B.; Kuhnle, A.; Lanza, G. Reinforcement learning for an intelligent and autonomous production control of complex job-shops under time constraints. *Prod. Eng.* **2020**, *14*, 319–328. [\[CrossRef\]](#)
94. Zhao, Y.; Zhang, H. Application of machine learning and rule scheduling in a job-shop production control system. *Int. J. Simul. Model* **2021**, *20*, 410–421. [\[CrossRef\]](#)
95. Kuhnle, A.; Kaiser, J.P.; Theiß, F.; Stricker, N.; Lanza, G. Designing an adaptive production control system using reinforcement learning. *J. Intell. Manuf.* **2021**, *32*, 855–876. [\[CrossRef\]](#)
96. Panzer, M.; Bender, B.; Gronau, N. Deep reinforcement learning in production planning and control: A systematic literature review. In Proceedings of the Conference on Production Systems and Logistics, Online, 10–11 August 2021.
97. Wojakowski, P.; Warzolek, D. The classification of scheduling problems under production uncertainty. *Res. Logist. Prod.* **2014**, *4*, 245–256.
98. Blackstone, J.H.; Phillips, D.T.; Hogg, G.L. A state-of-the-art survey of dispatching rules for manufacturing job shop operations. *Int. J. Prod. Res.* **1982**, *20*, 27–45. [\[CrossRef\]](#)
99. Blazewicz, J.; Ecker, K.H.; Pesch, E.; Schmidt, G.; Weglarz, J. *Handbook on Scheduling: From Theory to Applications*; Springer: Berlin/Heidelberg, Germany, 2007.
100. Ivanov, D.; Dolgui, A.; Sokolov, B. A dynamic approach to multi-stage job shop scheduling in an industry 4.0-based flexible assembly system. In *Advances in Production Management Systems. The Path to Intelligent, Collaborative and Sustainable Manufacturing: IFIP WG 5.7 International Conference, APMS 2017, Hamburg, Germany, 3–7 September 2017, Proceedings, Part I*; Springer International Publishing: Cham, Switzerland; pp. 475–482.
101. Modrak, V. (Ed.) *Mass Customized Manufacturing: Theoretical Concepts and Practical Approaches*; CRC Press: Boca Raton, FL, USA, 2017.
102. Komaki, G.M.; Sheikh, S.; Malakooti, B. Flow shop scheduling problems with assembly operations: A review and new trends. *Int. J. Prod. Res.* **2019**, *57*, 2926–2955. [\[CrossRef\]](#)
103. Yang, Y.; Li, X. A knowledge-driven constructive heuristic algorithm for the distributed assembly blocking flow shop scheduling problem. *Expert Syst. Appl.* **2022**, *202*, 117269. [\[CrossRef\]](#)

104. Nasiri, M.M.; Yazdanparast, R.; Jolai, F. A simulation optimisation approach for real-time scheduling in an open shop environment using a composite dispatching rule. *Int. J. Comput. Integr. Manuf.* **2017**, *30*, 1239–1252. [[CrossRef](#)]
105. Abdelmaguid, T.F. Bi-objective dynamic multiprocessor open shop scheduling for maintenance and healthcare diagnostics. *Expert Syst. Appl.* **2021**, *186*, 115777. [[CrossRef](#)]
106. Tremblet, D.; Thevenin, S.; Dolgui, A. Makespan estimation in a flexible job-shop scheduling environment using machine learning. *Int. J. Prod. Res.* **2024**, *62*, 3654–3670. [[CrossRef](#)]
107. Fu, Y.; Hou, Y.; Wang, Z.; Wu, X.; Gao, K.; Wang, L. Distributed scheduling problems in intelligent manufacturing systems. *Tsinghua Sci. Technol.* **2021**, *26*, 625–645. [[CrossRef](#)]
108. Zhang, Y.; Zhang, X.; Ding, Y. A Reinforcement Learning-Based Approach to Stochastic Job Shop Scheduling. *IEEE Trans. Autom. Sci. Eng.* **2020**, *17*, 72–83.
109. Rinciog, A.; Meyer, A. Towards standardizing reinforcement learning approaches for stochastic production scheduling. *arXiv* **2021**, arXiv:2104.08196.
110. Zeng, B.; Yang, Y. A Hybrid Reinforcement Learning and Fuzzy Logic Approach for Job Shop Scheduling. *J. Intell. Manuf.* **2017**, *28*, 1189–1201.
111. Zhang, G.; Huang, Y. Fuzzy reinforcement learning for multi-objective dynamic scheduling of a flexible manufacturing system. *J. Intell. Manuf.* **2005**, *16*, 293–304.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.



Article

Inversion-Based Deblending in Common Midpoint Domain Using Time Domain High-Resolution Radon

Kai Zhuang ¹, Daniel Trad ^{1,*} and Amr Ibrahim ²¹ Department of Earth, Energy, and Environment, Faculty of Science, University of Calgary, Calgary, AB T2N 1N4, Canada; kzhuang@ucalgary.ca² Department of Physics, Faculty of Science, Beni-Suef University, Beni-Suef 62521, Egypt; amr.ibrahim@ucalgary.ca

* Correspondence: daniel.trad@ucalgary.ca

Abstract: We implement an inversion-based deblending method in the common midpoint gathers (CMP) as an alternative to the standard common receiver gather (CRG) domain methods. The primary advantage of deblending in the CMP domain is that reflections from dipping layers are centred around zero offsets. As a result, CMP gathers exhibit a simpler structure compared to common receiver gathers (CRGs), where these reflections are apex-shifted. Consequently, we can employ a zero-offset hyperbolic Radon operator to process CMP gathers. This operator is a computationally more efficient alternative to the apex-shifted hyperbolic Radon required for processing CRG gathers. Sparse transforms, such as the Radon transform, can stack reflections and produce sparse models capable of separating blended sources. We utilize the Radon operator to develop an inversion-based deblending framework that incorporates a sparse model constraint. The inclusion of a sparsity constraint in the inversion process enhances the focusing of the transform and improves data recovery. Inversion-based deblending enables us to account for all observed data by incorporating the blending operator into the cost function. Our synthetic and field data examples demonstrate that inversion-based deblending in the CMP domain can effectively separate blended sources.

Keywords: deblending; sparse radon; inversion



Citation: Zhuang, K.; Trad, D.; Ibrahim, A. Inversion-Based Deblending in Common Midpoint Domain Using Time Domain High-Resolution Radon. *Algorithms* **2024**, *17*, 344. <https://doi.org/10.3390/a17080344>

Academic Editor: Frank Werner

Received: 16 May 2024

Revised: 25 June 2024

Accepted: 5 August 2024

Published: 7 August 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The quest for efficient seismic data acquisition is a well-known challenge in geophysical exploration. Simultaneous acquisition, also known as blended acquisition, stands as a solution, offering a method that significantly reduces the time spent in the field by concurrently firing multiple sources. This technique, which was initially proposed by Garottu [1] for vibroseis acquisition and later expanded by Beasley et al. [2], Beasley [3] for marine environments, allows for the interference of sources at acquisition time which are subsequently separated during processing. Despite its time-saving advantages, existing models and algorithms for deblending such data often come with limitations.

This study is motivated by the need to address these limitations and to enhance the accuracy and efficiency of seismic data deblending. Current methods, including random firing and phase encoding, have paved the way for approaches to data acquisition. However, the deblending process remains a significant bottleneck, with two predominant schools of thought: the denoising-based approach [4], which focuses on erratic and random noise attenuation, and the inversion-based approach [5], which seeks to reconstruct the signal by inverting the blending process. This paper will mainly focus on processing data from the random firing approach.

The introduction of small random delays to the source firing times, as suggested by Berkhout [6], is a critical step in the deblending process. Pseudo-deblending involves remapping the blended shots to zero firing time and duplicating frames according to the number of simultaneous sources. This manipulation due to the random delays renders

the interference from neighbouring sources as random noise in some of the common data domains, leaving only the target signal for recovery as coherent events. Pseudo-deblended data then show incoherent structures within the common midpoint gather (CMP), common receiver gather (CRG), and common offset gather (COF) domains, thus facilitating the deblending process. There are various methods to exploit the incoherency of blending noise, such as the separation work by Lin and Herrmann [7] and subsequent techniques such as Stolt Radon denoising [8], SSA filtering [9], denoising using a space-varying median filter [10], preconditioning with a discrete curvelet filter [11], and preconditioning with robust thresholding in the FK domain [12], each with their own advantages and drawbacks, often struggling with either issues in effectively distinguishing between signal and noise (source interference) for the denoising approaches or highly expensive and complex transform operators for the inversion-based approaches.

This paper introduces an inversion-based deblending approach utilizing the CMP domain, which promises a high-speed solution using a Radon transform operator. The utilization of the hyperbolic Radon transform for deblending serves multiple functions and significantly aids in the recovery of the seismic signal. Given the intrinsic nature of seismic acquisition, the acoustic responses emanating from the Earth's interior manifest as hyperbolic arrivals at the surface. The hyperbolic Radon transform, by aggregating data based on hyperbolic trajectories, is ideally suited for mapping seismic data. Opting for the CMP domain over the traditional CRG domain method allows for the employment of a more streamlined version of the Radon transform operator, resulting in reduced runtimes.

2. Theory

2.1. Blending Operator

The blending operator allows us to transition between the blended domain, where sources are overlapped, referred to as a super shot, to the unblended domain, where sources belong to their respective shot records with firing times adjusted to zero for each shot. This can be represented as the blending operator Γ , which contains the overlap and delay time information. We can then represent our unblended data cube as $\mathbf{D}$ and our blended data as $\mathbf{b}$ with the following relationship [6,13]:

$$\mathbf{b} = \Gamma \mathbf{D}. \quad (1)$$

Applying the adjoint of the blending operator to the blended data, we obtain the pseudo-deblended data, in the unblended space, with each shot occupying its own data frame as opposed to multiple shots per data frame. This can be expressed by the following [6,13]:

$$\tilde{\mathbf{D}} = \Gamma^H \mathbf{b}. \quad (2)$$

Here, $\tilde{\mathbf{D}}$ represents the pseudo-deblended data in standard shot frames. The blending matrix's role is to amalgamate shots with random delay times, effectively superimposing individual shot record data. This also means that pseudo-deblending acts to redistribute the data to their respective frames. This forms an overcomplete representation of the data with redundant information stored in each pseudo-blended source panel. It is important to note that the adjoint operator does not eliminate source interference from the data; it merely copies it around.

The successful deblending of data using the random firing technique hinges on the inclusion of random delay times. These delays are critical in converting source interferences into what resembles random noise when remapping the data via the adjoint blending operator. It is essential that these delay times are sufficiently randomized to the point that no patterns emerge in the remapped data domains, as coherent patterns in the delay times reduce the effectiveness of our coherency constraint. Furthermore, the delays should be sufficiently long to ensure the separability of events, adhering to the minimum delay length determined by a function of the lowest frequency and blending fold, as detailed in Jiang

and Abma [14]. Concurrently, they must be short enough that we gain an advantage in reduced acquisition times.

Random delays introduced into the blending schedule, and consequently the blending operator, are crucial for simplifying deblending. When the adjoint blending operator (pseudo-deblending) is applied to the blended dataset, the random delays become pivotal. Blending can be conceptualized as an operation on standard acquisition data that introduces source interference.

The influence of the addition on the random delay times can be seen when data are sorted into one of the other geometric domains after pseudo-deblending. After the application of the adjoint blending operator, coherent events can be seen in addition to the incoherent ones seen before the transform is applied; these coherent events are what we aim to extract, as they represent the part of the data that have their delay times properly corrected for, with the spontaneous data due to interfering shots that need to be removed. Through the use of an inversion method and a coherency constraint such as the high-resolution Radon transform, we can effectively remove the interfering signal and deblend our data.

2.2. Radon Transform

To solve the underdetermined issue seen in the inversion problem, we utilize the hyperbolic Radon transform (HRT) to further constrain the problem. This composite operator, which is made up of both the blending operator and the Radon transform operator, now reformulates the inversion into a sparsely constrained problem. The HRT is a variant of the Radon transform that sums across hyperbolic basis functions; in geophysics, this equation is defined as a function of arrival times and source–receiver offset and interval velocity, as seen in Thorson and Claerbout [15]:

$$m(p, \tau) = \int_{h_1}^{h_2} d(h, t = \sqrt{\tau^2 + p^2 h^2}) dh. \quad (3)$$

Here, $m(p, \tau)$ represents the Radon space data, p denotes the slowness, τ is the two-way travel time, h_1 and h_2 are the lower and upper offset limits, respectively, and d is the data space to be transformed. The slowness p is defined as the inverse of velocity $1/v$. Mapping based on hyperbolic arrivals allows for a precise and sparse representation of seismic data as seismic signals show up as hyperbolic events on standard seismic surveys. The Radon transform has been employed in various applications, including velocity analysis [15], multiple suppression [16], interpolation in the frequency or time domain [17], and deblending via denoising [8].

Although the HRT is well suited for seismic data, it has several limitations. Primarily, the transform requires the hyperbolic arrivals to be symmetric and centred in the data space, which is not necessarily the case in real-world data. In reality, the dipping of subsurface structures causes hyperbolic arrivals to shift up-dip, resulting in non-centred arrivals. Symmetry is challenged by two factors: anisotropy and survey conditions. Surface sensor arrays are generally not placed on a perfectly flat surface, leading to deviations from hyperbolic arrivals. This issue can be easily resolved through a trace statics correction, which is part of the standard processing flow. For the anisotropy issue, a modified equation can be used to account for curve stretching in simple cases. The shift from centred locations presents a somewhat complex problem with multiple potential solutions. The first solution involves the implementation of an apex-shifted hyperbolic Radon transform (ASHRT) [18]. In this approach, multiple Radon panels are generated for a single data image, each corresponding to a series of expected shifted apex locations. While this solution is robust and maps well to seismic data, it significantly increases runtimes as there is no shortcut to generating Radon panels, and processing times are multiplied by the number of apexes scanned. The second solution to the centring problem involves transforming the data into the CMP domain, where apexes are centred even for dipping reflections. However, this

solution has a major drawback: the CMP domain generally has worse sampling than other data domains, making aliasing a significant concern.

2.3. Inversion-Based Deblending

Given our already defined transformation operator, the mapping operator Γ that maps the data space $\mathbf{b}$ to the model space $\mathbf{D}$ in Equation (1), we can frame the problem as an inversion problem that reverses the operation [19]. Inverting this operation, we need to formulate an objective function that minimizes the misfit between the data and the modelled data, in other words, minimizing the model information that does not map to the data. This objective function is often presented as follows [17]:

$$\begin{aligned} & \text{minimize } \|\mathbf{D}\|_2^2 \\ & \text{Subject to } \|(\mathbf{b} - \Gamma\mathbf{D})\|_2^2. \end{aligned} \quad (4)$$

With the objective function for the inversion of blended data being written as

$$\mathbf{J} = \|\mathbf{b} - \Gamma\mathbf{D}\|, \quad (5)$$

Direct inversion of the blending operator is ill-posed; thus, no unique solution can be assessed by inverting just the blending operator Γ . This ill-posed nature is due to the fact that the blending operator only serves to collect and shift data, while the adjoint operator serves only to distribute it. To circumvent this problem, we reformulated the inversion scheme by adding the Radon operator as a coherency constraint.

$$\mathbf{J} = \|\mathbf{b} - \Gamma\mathbf{L}\mathbf{m}\| + \mu\|\mathbf{m}\|, \quad (6)$$

where $\mathbf{L}$ is the Radon transform. By formulating the problem this way, we transition from an ill-posed problem to one that is sparse constrained.

2.4. Sparse Regularization

We utilize sparse regularization for our inversion scheme as a way to solve for the ℓ_1 norm in the model space, as a more robust solution to the Radon transform [17] and, in turn, the deblending scheme. To solve for the sparse solution in this paper, we utilize the iteratively reweighted least-squares inversion (IRLS) method [20]. IRLS is a popular way to solve for sparse and/or robust inverse solutions due to its simplicity. Effectively, IRLS reformulates the calculation of the least absolute value solution as a repetition of least-squares iterations using weights in the data and model spaces. This approach is more efficient than directly solving for the least absolute value solution as we can utilize faster least squares solvers like the conjugate gradient to reduce computational times. Utilizing IRLS also allows for mixed norm inversions; for this paper, we use an ℓ_2 -norm in the data space and an ℓ_1 -norm in the model space.

As many researchers have explored, applying a sparse constraint to the Radon transform offers numerous benefits and few drawbacks for the mapping of seismic data [17]. These improvements address issues such as missing data due to aliasing and limited aperture information in mapping and reconstruction. The advantages also extend to the use of the Radon transform in deblending. The primary benefit of enforcing a sparse constraint in the model space for our method is that, as a path integral, the erratic and random noise from the random delay times maps poorly in Radon space. This poor mapping results in low-amplitude smears in Radon space, in contrast to the high-amplitude focal points seen from coherent hyperbolic arrivals. This is complemented by the fact that in the shot frames where the data belong, the signals sum and focus correctly. Consequently, the solver will converge on the sparsest solution by downweighting smeared blended noise and upweighting coherent focal points. This is also the most deblended solution, as the data are remapped into the frames where they appear most coherent, and blending noise is fully explained by the mixing of neighbouring shots, effectively deblending the data.

IRLS exists as a way to indirectly solve the sparse solution of an inversion problem by framing it as multiple iterations of a least squares problem, in this case, conjugate gradient (CG), with the outer iterations updating data and model weights to have the CG algorithm solved, driving the solution to the ℓ_1 -norm. Enforcing ℓ_1 -norm in the data space allows for reduced sensitivity to outliers which would otherwise negatively impact the inversion, often referred to as robust inversion. Enforcing ℓ_1 -norm in the model space solves for the sparsest solution, mainly the model that explains the data with the smallest amount of data points; this is referred to as the sparse solution. The utilization of IRLS allows us to mix and match which norms we wish to utilize through the calculation of the weights and, thus, are able to solve for a mixed norm solution.

By adding model and data weights, we can then dictate the algorithm to converge towards the least absolute value solution (ℓ_1) or least squares (ℓ_2) by reformulating the original least-squares equation through the minimization of the modified equation:

$$\begin{aligned} &\text{Minimize } ||\mathbf{W}_m \mathbf{m}||_2^2 \\ &\text{Subject to } ||\mathbf{W}_r(\mathbf{b} - \mathbf{\Gamma L m})||_2^2. \end{aligned} \quad (7)$$

where $\mathbf{W}_r$ is the data weight operator and $\mathbf{W}_m$ is the model weight operator. These weights allow the user to adjust the preference to either the residual or model, where $\mathbf{W}_m$ can be changed to obtain a certain fit to either a sparse ℓ_1 or smooth ℓ_2 model, and $\mathbf{W}_r$ can be changed to prefer non-spontaneous events or “weight” all events equally in the input data. The operators are usually diagonal matrices. For example, $\mathbf{W}_r$ can be constructed as follows:

$$\begin{aligned} \text{diag}(\mathbf{W}_r) &= |\mathbf{r}|^{(p-2)/2} \\ \text{where } \mathbf{r} &= \mathbf{\Gamma L m} - \mathbf{b}. \end{aligned} \quad (8)$$

where p is either 1 or 2 for measuring the model size by using either the ℓ_1 or ℓ_2 data norm. $\mathbf{W}_m$ is normally constructed as

$$\text{diag}(\mathbf{W}_m) = |\mathbf{m}|^{(2-p)/2}. \quad (9)$$

where p is either 1 or 2 for inverting based on either the ℓ_1 or ℓ_2 model norm. Due to the data weights being created through division, a damping factor is chosen to avoid division by zero. The damping factor is determined as a percentile of the data. The main advantage of the inversion approach for deblending is that all of the signal and blending noise can be explained in other domains where they appear coherent, as opposed to the denoising approach, which seeks to remove the blending noise frame by frame.

We can also use the sparse inversion to demonstrate the denoising approach to deblending using the Radon transform to denoise individual receiver gathers. The denoising approach first applies pseudo-deblending to the blended data to return a dataset in standard acquisition frames, seen in Figure 1b, which then allows for the separation of the shots through denoising in the other domains. The general denoising process using sparse/robust Radon first applies the adjoint blending operator to the blended data, $\mathbf{b}$:

$$\tilde{\mathbf{D}} = \mathbf{b} \mathbf{\Gamma}^H, \quad (10)$$

Then the pseudo-deblended data are used in the inversion with the following objective function to denoise using the Radon transform:

$$||\mathbf{W}_r(\tilde{\mathbf{D}} - \mathbf{R m})||_p^p + \mu ||\mathbf{W}_m \mathbf{m}||_q^q, \quad (11)$$

where $\tilde{\mathbf{D}}$ is the pseudo deblended data, $\mathbf{R}$ is the Radon transform operator, $\mathbf{m}$ is the model, μ is the trade-off parameter, and p and q are the data and model norms, respectively. A graphical representation of Radon denoising can be seen in Figure 2, using standard

hyperbolic Radon to downweight spontaneous events in the shot record in an attempt to recover the coherent signal.

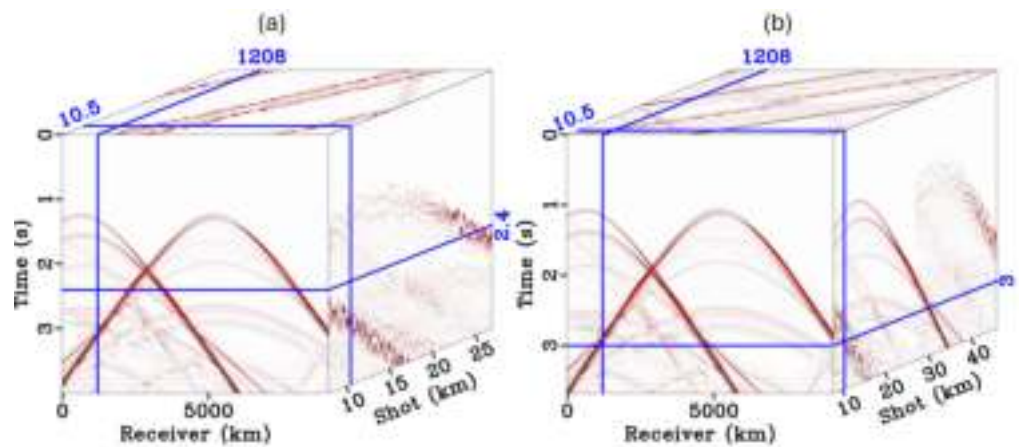


Figure 1. Application of the adjoint operator: (a,b) is the result; it can be seen in other geometrical domains that our target is coherent while interfering signals remain incoherent.

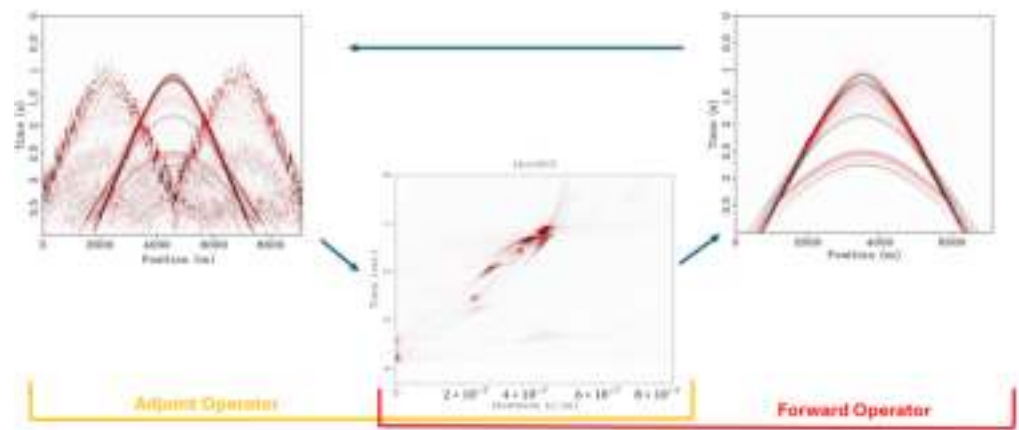


Figure 2. Denoising approach to deblending: The denoising approach to deblending uses sparse/robust Radon to remove the blending noise from each pseudo-deblended receiver frame.

Given that blending noise from one shot record is a signal from another, our sporadic blending noise has the same amplitudes as that of the shot itself. This issue at high blending fold manifests in the high-amplitude noise completely overwhelming sections of low-amplitude events in the signal, which makes traditional filtering-based recovery methods nearly impossible. However, the use of inversion allows us to explain the interfering high-amplitude events and effectively remap only the interfering amplitude, thus allowing us to recover the lower-amplitude signals. The schematic for the inversion-based approach can be seen in Figure 3. However, this approach does not come without its own drawbacks, mainly in the memory requirements, as the entire blending dataset plus the corresponding model must be held in memory. Despite initial expectations, this method does not have any significant runtime difference compared to a similar inversion-based denoising approach, as an inversion must be performed in both cases. When compared to a Radon denoising approach, the inversion approach uses a different objective function as it seeks to incorporate both the blending operator and the Radon operator in the inversion. We can formulate the objective function using the already discussed weights as well as Equation (6):

$$J = ||\mathbf{W}_r(\mathbf{b} - \Gamma \mathbf{L} \mathbf{m})||_p^p + \mu ||\mathbf{W}_m \mathbf{m}||_q^q. \quad (12)$$

Here, the input is not the pseudo-deblended data, but the blended data, and the operator contains both the blending and Radon operators.

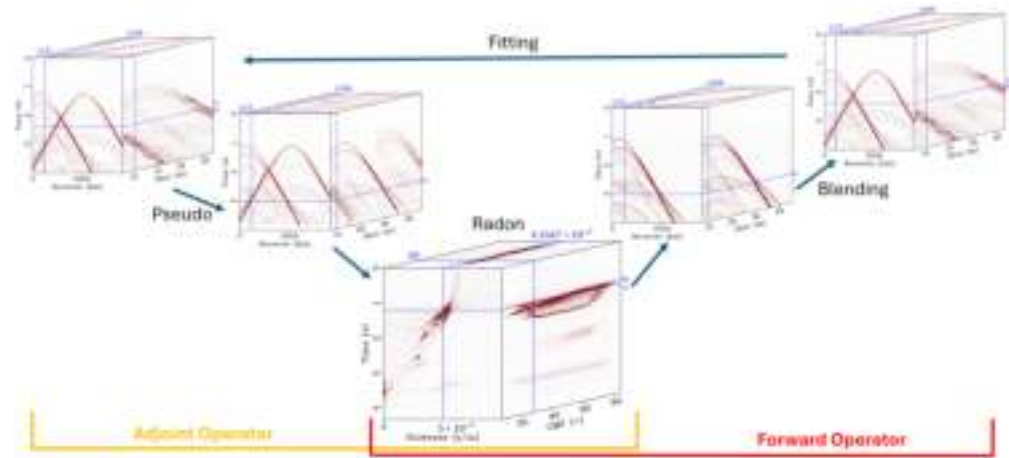


Figure 3. Schematic of inversion-based deblending: Combining the blending operator with the Radon transform allows us to remap sporadic blending interference back into the coherent signal in its proper location, effectively explaining, rather than removing, interference.

By implementing deblending as an inversion-based operation using the blending and Radon operators together, the system acts like an overcomplete dictionary, where the information is represented and redundant in the pseudo-deblended frames. This aids in convergence as the redundant information explains each other through the remapping of data back to their respective shot frames.

3. High-Performance Computing (HPC) Considerations

The sparse hyperbolic Radon transform is used in our deblending algorithm to reformat the ill-posed blending operator to an inversion problem that uses sparse regularization. This Radon operator allows us to benchmark the performance of a highly parallelizable operation using a series of parallel processing methods. To properly code the HRT to run efficiently in openMP, we send outer offset loops to separate threads to calculate the transform in the adjoint operator; for the forward operator, we then send each outer slowness loop instead. The pseudo-code for the Radon transform can be written as seen in Algorithm 1.

Algorithm 1 Radon Pseudo Code

```

1: function RADON
2:   #PRAGMA OMP PARALLEL FOR                                ▷ insert for parallelization of loop
3:   for q = slowness do
4:     for h = offset do
5:        $moveout = h^2 * q$ 
6:       for it = 0 to nt do
7:          $time = \sqrt{(it * dt)^2 + moveout}$ 
8:          $model[iqNt + it] += data[ihNt + INT(time/dt)]$ 
9:       end for
10:    end for
11:  end for
12: end function

```

For our implementation in CUDA, we set up a two-dimensional thread configuration with one dimension for each of the time and slowness dimensions in the standard code, then calculated the transform independently for each output cell for each thread. To implement the HRT in openMPI, we send the data pertaining to each shot in the transform to separate

nodes and then calculate the transform independently on each node using either the CUDA or openMP methods stated above. We then collect the data at the master at the end of the calculation. The CUDA pseudo-code can be seen in Algorithm 2.

Algorithm 2 CUDA Radon Pseudo Code

```

1: __global__
2: function CUDA_RADON
3:   int it = blockIdx.x * blockDim.x + threadIdx.x
4:   int iq = blockIdx.y * blockDim.y + threadIdx.y
5:   if (iq >= nq || it >= nt) return
6:   int iqNt = iq * nt
7:   double timemax = dt * nt
8:   for h=offset do
9:     int ihNt = ih * nt
10:    double moveout =  $h^2 * q$ 
11:    double time =  $\sqrt{(it * dt)^2 + moveout}$ 
12:    model[iqNt + it] += data[ihNt + (INT)(time/dt)]
13:   end for
14: end function

```

Because a significant number of parallel applications use multidimensional data, the CUDA API groups thread blocks into multiple dimensions, up to three. This is beneficial to the implementation of Radon as there are generally three data dimensions for the transform, one each for offset, time, and velocity. However by assigning each of our variables to a dimension, we can eliminate their respective loops, but this is not the most efficient way to implement the algorithm. Because the model is an accumulation across the offset dimension, we would encounter a data race if the offset dimension h is executed in parallel. If the offset were set as a third thread dimension in CUDA, an atomic operator would be required; this was initially tested and produced significantly slower results than the algorithm provided above. By only parallelizing the two grid dimensions, we avoid the need to use atomic accumulation operators, which can cause significant slowdowns, with each thread performing work for each grid cell in what can be called output-aligned optimization.

This algorithm is interesting in that most of it can be optimized in parallel, but there exist many challenges to proper parallelization. The main challenge of optimizing and parallelizing the algorithm is due to the large memory swaps required for every iteration. These memory issues arise due to the constant need to change domains in the blending operator, resulting in large-scale non-sequential access to the data. The other challenge we face is that this algorithm cannot be properly parallelized across distributed memory systems (servers) using openMPI, as the gradient calculation at every iteration needs to be calculated using the blended operator which requires resorting all data. Since distributed memory in clusters requires splitting the dataset across nodes instead of keeping it all in the master node's RAM, the resorting process imposes a very difficult challenge for an openMPI implementation. Due to this requirement, each node in a cluster would need to send data back after every calculation to calculate the next step, and then the master will need to resend the results back to each node. For our implementation in CUDA instead of using multiple nodes, we utilize a single high-performance machine, where both the Radon and the least-squares descent algorithm are rewritten using CUDA kernels.

4. Results

As our test data are mainly synthetic and marine data, we assume that the data space is free of outliers; thus, we utilized the ℓ_2 in the data space, for a mixed norm ℓ_2 - ℓ_1 inversion. For our testing, we utilized three datasets, two that were created synthetically through finite difference, and one real-world marine dataset from the Gulf of Mexico that was numerically blended. For the synthetic datasets, finite difference modelling was run with 300 shots and 300 receivers evenly spaced at the surface. The first dataset and the most simplistic

is the wedge model dataset seen in Figure 4. The wedge model consists of two layers, one triangular wedge, and a set of hard points to test the reconstruction of low-amplitude diffractions. The wedge data were blended with two overlapping shots with delay times randomized between 0 and 400 samples.

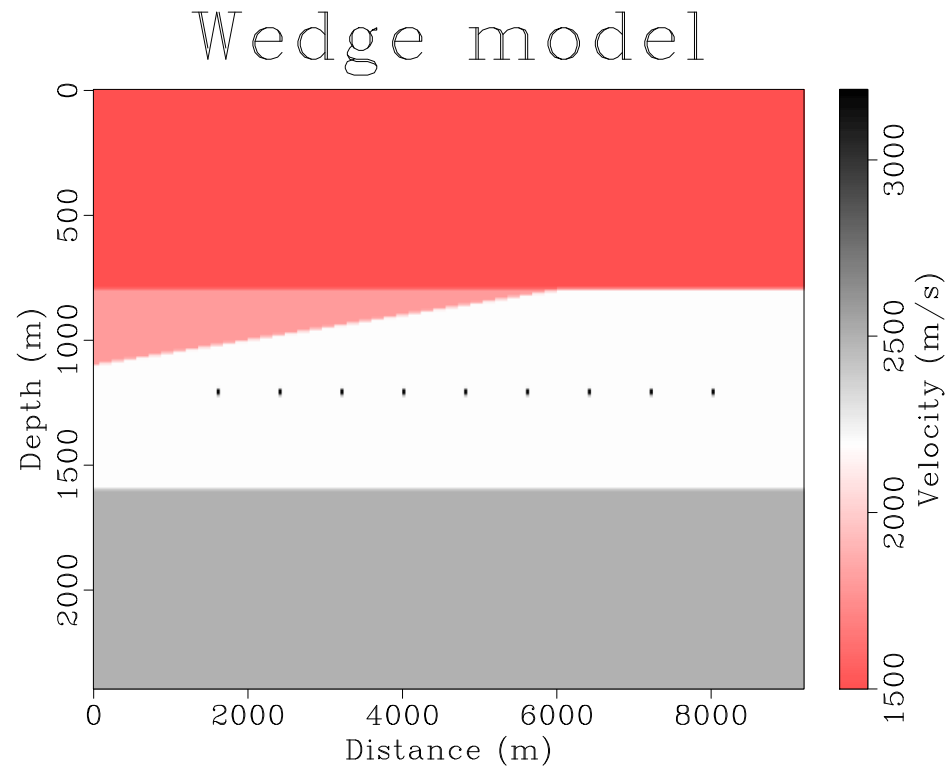


Figure 4. Velocity model 1: Wedge model consisting of two layer 9 diffraction points and one wedge.

The deblending can be seen in Figure 5, where the blending noise seen in the pseudo deblended data in Figure 5b is removed through inversion in Figure 5c. Subsequently, the results were transformed into the shot domain, as depicted in Figure 6.

Despite some residual interference in the blended shot, the deblending process effectively removed the secondary interfering shot and retained the target shot well, including the diffractions. These can be observed near the top of both Figure 6b,c.

A comparison of our inversion method and a denoising method is shown in Figure 7. The denoising method also utilizes the sparse HRT as a denoising engine to remove the random blending noise in the pseudo-deblended gathers. The primary distinction from the inversion method is that the denoising approach does not involve the blending operator. In the CMP domain, the high amplitudes of the blending noise make direct removal through general noise reduction methods difficult. This is because differentiating noise from data becomes increasingly difficult as the blending noise has the same amplitude as the signal we aim to recover. Without the additional information that the blending operator provides, this similarity makes it challenging to discern which parts of the data are noise and which are coherent signals. As can be seen in Figure 7, the denoising result successfully deblended the data, but at the cost of the lower-amplitude data of the diffraction arrivals, which were removed along with the blending noise.

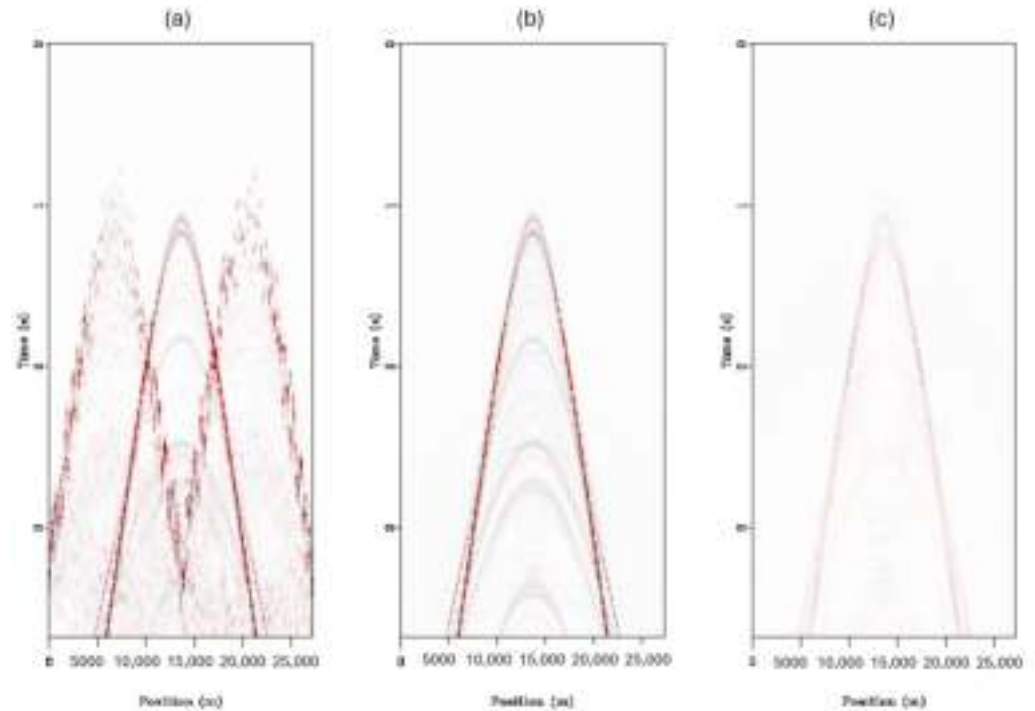


Figure 5. Wedge model test results (CMP): Deblending test 1 in CMP domain with (a) the pseudo-deblended data, (b) the deblended data, and (c) the reconstruction error shown, respectively.

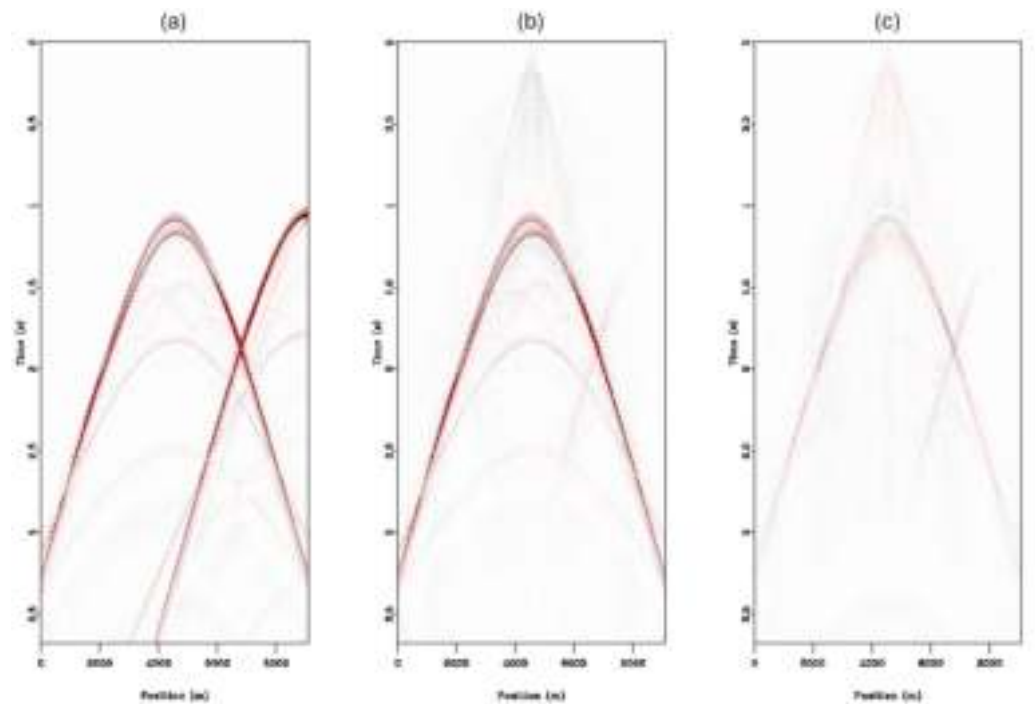


Figure 6. Wedge model test results (shot domain): Deblending test 1 in shot domain with (a) the pseudo-deblended data, (b) the deblended data, and (c) the reconstruction error shown, respectively.

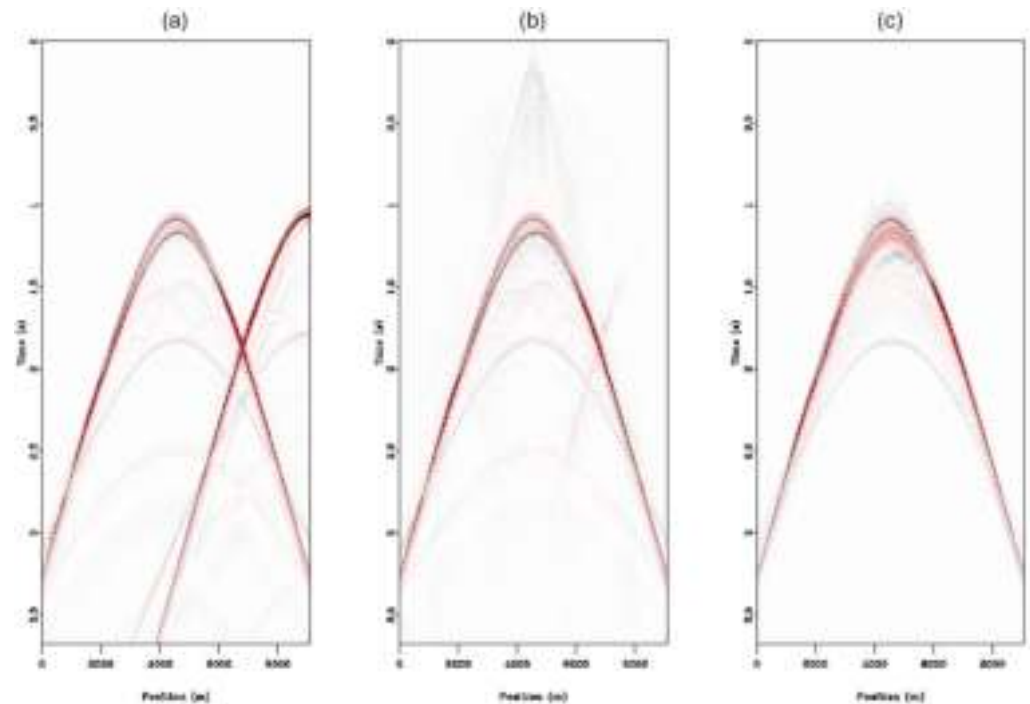


Figure 7. Comparison of wedge model results between inversion and denoise approach: inversion and denoising approaches in shot domain, with (a) blended, (b) inversion debblended, and (c) denoise debblended.

Synthetic test number 2 was generated via finite difference using the standard Marmousi model [21]. As test 1 showed that our debblending engine effectively debblended our data, we decided to make test number 2 more of a stress test, one in which a non-inversion-based debblending method would have difficulty effectively debblending. We significantly increased the difficulty by reducing the range of our random delay times and also increasing the number of overlapping shots to five up from two, conditions we do not expect to see in the real world currently. Figure 8a shows the blended data in the CMP domain; it is easy to visualize how difficult it would be for a normal denoiser without the blending operator to distinguish the data from the interference in this image.

Analyzing the results of the second test, we can see in Figure 8 that our recovery of low-amplitude reflections is effective even when completely overwhelmed by high-amplitude interference from other shots. Taking a look at the reconstruction error for the shot domain in Figure 9, we can see that the main differences are reduced amplitudes of the higher-amplitude events and the introduction of very low amplitude background noise; we believe that the amplitude error may come from enforcing too much sparsity in the model space, which generally causes these issues in high-resolution Radon, or not fully separating the blending shots and smearing the higher amplitudes across the data as background noise.

Our final test was a real-world dataset pulled from the Gulf of Mexico, known as the BP Tiber WATS dataset [8,22], which was then numerically blended. The blending scheme was configured differently compared to the others, as this was also a test of another acquisition schedule known as continuous listening. In continuous listening, we effectively have a single supershot, as the data are recorded until all shots are fired and shots are allowed to overlap. The cut-down dataset consisted of 90 shots and 90 receivers, with a shot-to-shot overlap of 70% randomized between 0 and 200 samples. The results of the debblending of the Gulf of Mexico dataset can be seen in the CMP domain in Figure 10 and the shot domain in Figure 11. Analyzing the results, we can see that the debblending algorithm is exceptionally effective at shot separation on the real-world dataset, with good recovery of all events seen in the data. The main error we can observe is that in the shot

domain, shots are not perfectly deblended, and some amplitude from the trailing shot can still be seen in the recovered shot. This can be observed around 5.2 s in Figure 11c.

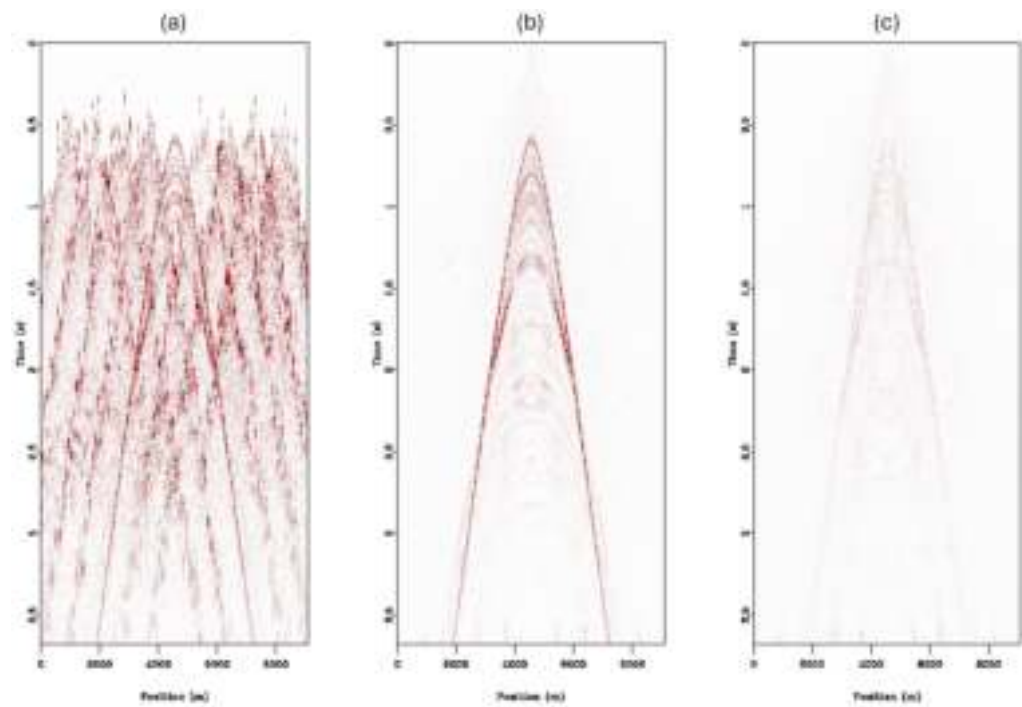


Figure 8. Marmousi model test results (CMP): Deblending test 2 with 5 shots, with (a) blended, (b) deblended, and (c) reconstruction error, respectively.

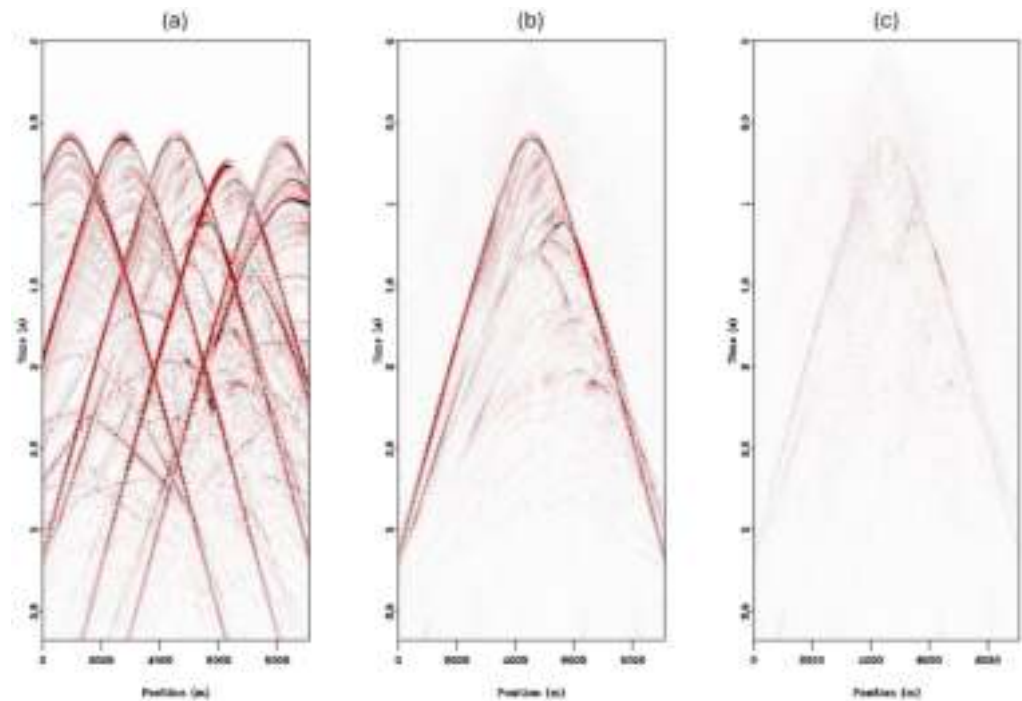


Figure 9. Marmousi model test results (Shot domain): Deblending test 2 with 5 shots, with (a) blended, (b) deblended, and (c) reconstruction error, respectively.

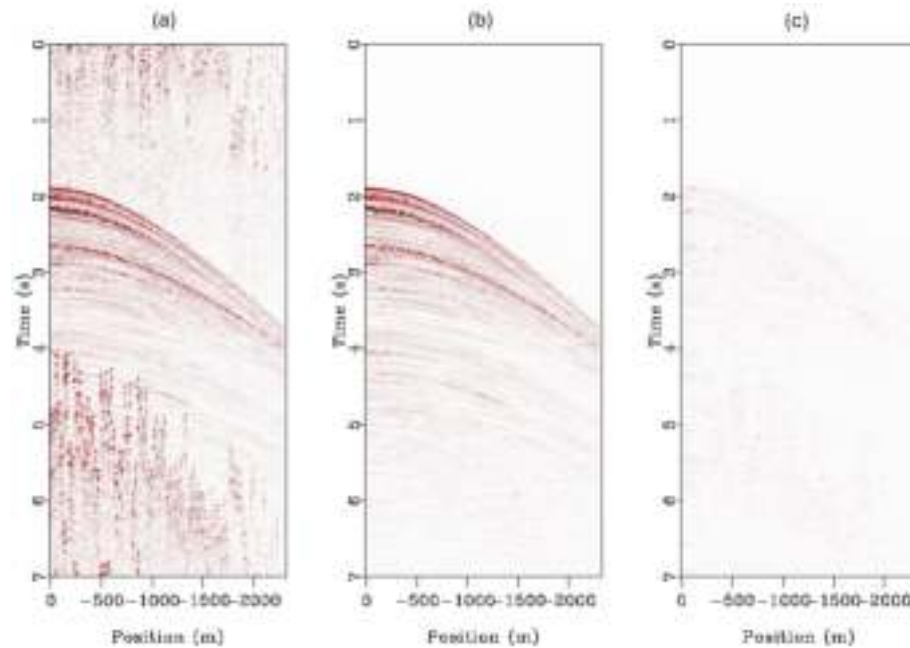


Figure 10. Gulf of Mexico data in CMP domain: Deblending using a Gulf of Mexico marine dataset with results in the CMP domain, with blended data in (a), debbled in (b), and the reconstruction error in (c).

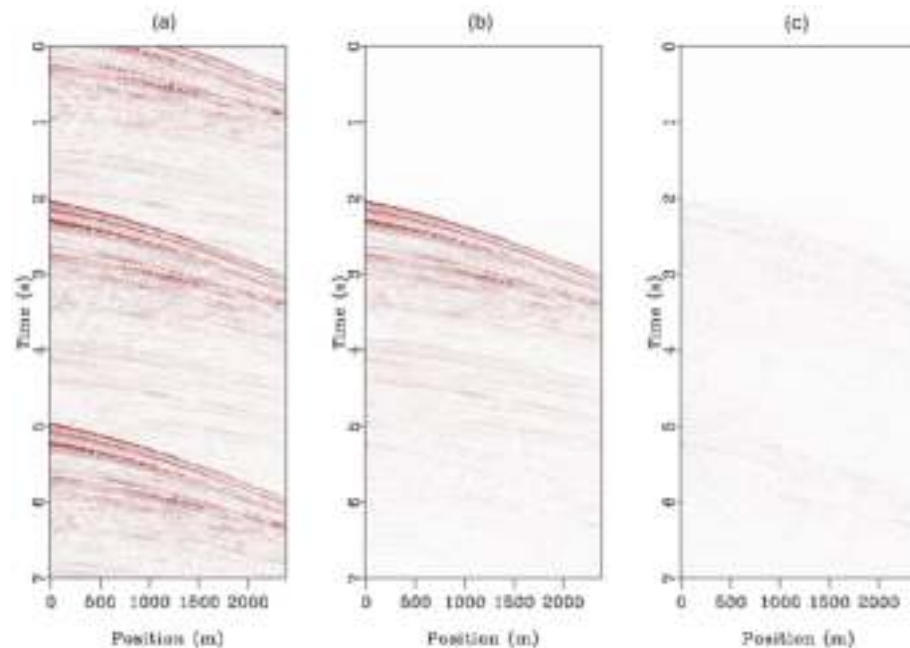


Figure 11. Gulf of Mexico data: Deblending using a Gulf of Mexico marine dataset with a shot time interference of 70%, with blended data in (a), debbled in (b), and reconstruction error in (c).

Limitations to this approach lie in the assumptions inherent in the use of the hyperbolic Radon transform. The main assumption of this approach is the hyperbolic trajectory of all seismic arrivals, which are expected to be accurately represented in Radon space. However, this presumption becomes problematic when considering the unique characteristics of land seismic data. The statics inherent to land data can induce vertical shifts in the samples, deviating from their hyperbolic alignment. Such discrepancies lead to an ineffective mapping in the Radon domain and potentially compromise the integrity of the deblending process.

We ran both the deblending algorithm and Radon transform algorithm on a range of different hardware to show the performance of the algorithms using the different parallelization methods in different conditions: a laptop, two workstations, and our in-house server cluster. The laptop utilizes an Intel i7 8750H 6-core CPU with an Nvidia RTX 2060 MAX-Q GPU, the first workstation is equipped with an AMD Threadripper 3960× 24-core CPU, with an Nvidia RTX 3080Ti, and the second workstation has an Intel i7 9800× 8-core CPU equipped with an Nvidia RTX 2060-super GPU. These results can be seen in Table 1. The Runtime between the Radon transform and the deblending algorithm cannot be directly compared as they were performing different tasks; the relative difference in the runtime between different hardware shows the effect the sorting has on the different implementations.

Table 1. Table of average runtimes for Radon and least squares deblending on different machines.

Processor (API)	Radon Transform Time (s)	LS Deblending Time (s)
i7 6800k 6-core (openMP)	4462	2590
i5 8600 6-core (openMP)	3773	1748
i7 8750H 6-core (openMP)	4044	2203
i7 9800× 8-core (openMP)	2715	1390
TR 3960× 24-core (openMP)	1123	581
RTX 3060 (CUDA)	147	174
RTX 3080Ti (CUDA)	108	142
RTX 2060 Max-Q (CUDA)	184	206
RTX 2060 Super (CUDA)	157	198
TR 3960× (openMPI)	420	—

It can be seen that there is a significant computational advantage in moving from CPU processing (openMP) to GPU-based processing (CUDA). The openMPI results show a significant speedup compared to the openMP results and close the gap with the CUDA results; this speedup can mainly be attributed to the coarser-grained nature of the openMPI. The openMPI implementation shares fewer resources between threads; thus, less overall memory management has to be performed by the system, at the cost of overall RAM space as more copies of data are made. In standard Radon transform testing, the CUDA-based code reduced compute time by over an order of magnitude compared to the openMP version; after the introduction of the least-squares algorithm, the advantage diminished by a large amount. Due to the openMPI distributing all work, including the LS optimization across the nodes, it can run the LS optimization in a parallel fashion that neither the openMP nor CUDA code can, resulting in an overall faster computational time versus openMP and approaches to the CUDA implementation. OpenMPI deblending, however, cannot be properly implemented as the host node must collect all data from the other nodes to perform the LS calculation every iteration; this results in large memory transfers across the network which incur a significant latency and transfer rate hit. The overhead associated with the openMPI implementation is so large that we did not notice a benefit in our prototyping of the program as compared to even single-threaded runtimes. It must be noted, however, that openMPI used the largest amount of RAM in this exercise, about 122 GB, compared to the openMP and CUDA implementations which used 8 GB of RAM and 8 GB + 8 GB (mirrored) VRAM, respectively. The main trade-off between a CUDA/GPU-based approach and an OpenMP approach would be that of speed versus portability; the CUDA approach is significantly faster but requires an Nvidia GPU, while the OpenMP approach will work on any device that supports OpenMP parallelization in the compiler, which is any modern CPU that exists on the market.

5. Conclusions

We proposed a method for deblending that employs a linear inversion of a composite transform operator in the CMP domain. This inversion, when used with the blending

operator, allows us to predict and remap, rather than eliminate, interfering shot information. This, in turn, facilitates the accurate recovery of low-amplitude events, even in data that are heavily blended. The implementation of a sparse constraint on the inversion algorithm enables us to arrive at a unique solution that enhances the coherency of the signal in hyperbolic arrivals. Furthermore, the use of hyperbolic Radon transforms in the CMP domain provides us with an operator that is efficient and well matched to our data, promoting coherence in our recovery and circumventing aliasing issues. Consequently, we are confident that our inversion approach is an effective strategy for deblending simultaneously sourced data while safeguarding low-amplitude events. We have also explored additional methods to accelerate this framework through applications on both the CPU and GPU, each with its own set of advantages and disadvantages.

Author Contributions: Conceptualization, K.Z. and D.T.; methodology, K.Z. and D.T.; software, K.Z.; validation, K.Z., D.T. and A.I.; formal analysis, K.Z.; investigation, K.Z.; data curation, K.Z. and A.I.; writing—original draft preparation, K.Z.; writing—review and editing, A.I. and D.T.; supervision, D.T. All authors have read and agreed to the published version of the manuscript.

Funding: We thank the sponsors of the Consortium for Research in Elastic Wave Exploration Seismology (CREWES) for their continued support. This work was funded by CREWES industrial sponsors and NSERC (Natural Science and Engineering Research Council of Canada) through the grant CRDPJ 543578-19.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

COF	Common offset
CMP	Common midpoint
CRG	Common receiver gather
HRT	Hyperbolic Radon transform
ASHRT	Apex-shifted hyperbolic Radon transform
CG	Conjugate gradient
IRLS	Iterative reweighted least squares
CUDA	Compute Unified Device Architecture (Nvidia programming API)
API	Application programming interface

References

1. Garottu, R. Simultaneous recording of several vibroseis seismic lines. In *SEG Technical Program Expanded Abstracts 1983*; Society of Exploration Geophysicists: Houston, TX, USA, 1983; pp. 308–310. [\[CrossRef\]](#)
2. Beasley, C.J.; Chambers, R.E.; Jiang, Z. A new look at simultaneous sources. In *SEG Technical Program Expanded Abstracts 1998*; Society of Exploration Geophysicists: Houston, TX, USA, 1998; pp. 133–135. [\[CrossRef\]](#)
3. Beasley, C.J. A new look at marine simultaneous sources. *Lead. Edge* **2008**, *27*, 914–917. [\[CrossRef\]](#)
4. Mahdad, A.; Doulgeris, P.; Blacquiére, G. Separation of blended data by iterative estimation and subtraction of blending interference noise. *Geophysics* **2011**, *76*, Q9–Q17. [\[CrossRef\]](#)
5. Abma, R.; Howe, D.; Foster, M.; Ahmed, I.; Tanis, M.; Zhang, Q.; Arogunmati, A.; Alexander, G. Independent simultaneous source acquisition and processing. *Geophysics* **2015**, *80*, WD37–WD44. [\[CrossRef\]](#)
6. Berkhout, A.J. Changing the mindset in seismic data acquisition. *Lead. Edge* **2008**, *27*, 924–938. [\[CrossRef\]](#)
7. Lin, T.; Herrmann, F. Designing Simultaneous Acquisitions with Compressive Sensing. In *Proceedings of the 71st EAGE Conference and Exhibition Incorporating SPE EUROPEC*, Amsterdam, The Netherlands, 8–11 June 2009; p. cp-127-00269. [\[CrossRef\]](#)
8. Ibrahim, A. Separating Simultaneous Seismic Sources Using Robust Inversion of Radon and Migration Operators. Ph.D. Thesis, University of Alberta, Edmonton, AB, Canada, 2015.
9. Liu, B.; Sacchi, M. Minimum weighted norm interpolation of seismic records. *Geophysics* **2004**, *69*, 1560–1568. [\[CrossRef\]](#)
10. Chen, Y. Deblending using a space-varying median filter. *Explor. Geophys.* **2015**, *46*, 332–341. [\[CrossRef\]](#)

11. Mahmoudian, F.; Curtis, J.; De Meersman, K.; Kennedy, R.; Staples, G. Deblending of Simultaneous Source Data. Available online: <https://geoconvention.com/wp-content/uploads/abstracts/2023/91390-deblending-of-simultaneous-source-.pdf> (accessed on 25 June 2024).
12. Stanton, A.; Wilkinson, K. Robust deblending of simultaneous source seismic data. *arXiv* **2018**, arXiv:1812.06040.
13. Berkhout, A.J.; Blacqui re, G.; Verschuur, D.J. The concept of double blending: Combining incoherent shooting with incoherent sensing. *Geophysics* **2009**, *74*, A59–A62. [[CrossRef](#)]
14. Jiang, Z.; Abma, R. An analysis on the simultaneous imaging of simultaneous source data. In *SEG Technical Program Expanded Abstracts 2010*; Society of Exploration Geophysicists: Houston, TX, USA, 2010; pp. 3115–3119. [[CrossRef](#)]
15. Thorson, J.R.; Claerbout, J.F. Velocity-stack and slant-stack stochastic inversion. *Geophysics* **1985**, *50*, 2727–2741. [[CrossRef](#)]
16. Foster, D.J.; Mosher, C.C. Suppression of multiple reflections using the Radon transform. *Geophysics* **1992**, *57*, 386–395. [[CrossRef](#)]
17. Trad, D.; Ulrych, T.; Sacchi, M. Latest views of the sparse Radon transform. *Geophysics* **2003**, *68*, 386–399. [[CrossRef](#)]
18. Trad, D.; Hargreaves, N.; verWest, B.; Wombell, R. Multiple Attenuation Using an Apex Shift Radon Transform. In *SEG Expanded Abstracts*; Society of Exploration Geophysicists: Houston, TX, USA, 2004. [[CrossRef](#)]
19. Claerbout, J.F. *Earth Soundings Analysis: Processing versus Inversion*; Blackwell Scientific Publications: London, UK, 1992.
20. Scales, J.A.; Gersztenkorn, A.; Treitel, S. Fast Ip solution of large, sparse, linear systems: Application to seismic travel time tomography. *J. Comput. Phys.* **1988**, *75*, 314–333. [[CrossRef](#)]
21. Brougois, A.; Bourget, M.; Lailly, P.; Poulet, M.; Ricarte, P.; Versteeg, R. Marmousi, model and data. In Proceedings of the EAEG workshop-practical aspects of seismic data inversion, Copenhagen, Denmark, 28 May–1 June 1990.
22. Naranjo, J.C.; Ekstrand, E.J.; Etgen, J.; Hargrove, K.L.; Brewton, J.G.; Garcia, O.; Astvatsaturov, G.; Hite, D.; Howieson, B.; Bai, B.; et al. Survey design and implementation of a multiple wide-azimuth towed streamer seismic acquisition strategy at the Tiber Discovery; deep water Gulf of Mexico, USA. In *SEG Technical Program Expanded Abstracts 2011*; Society of Exploration Geophysicists: Houston, TX, USA, 2011; pp. 132–136. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.

Article

Precedence Table Construction Algorithm for CFGs Regardless of Being OPGs

Leonardo Lizcano, Eduardo Angulo  and José Márquez * 

Department of Computer Science and Engineering, Universidad del Norte, Barranquilla 081007, Colombia; ldlizcano@uninorte.edu.co (L.L.); edangulo@uninorte.edu.co (E.A.)

* Correspondence: jmarquez@uninorte.edu.co

Abstract: Operator precedence grammars (OPG) are context-free grammars (CFG) that are characterized by the absence of two adjacent non-terminal symbols in the body of each production (right-hand side). Operator precedence languages (OPL) are deterministic and context-free. Three possible precedence relations between pairs of terminal symbols are established for these languages. Many CFGs are not OPGs because the operator precedence cannot be applied to them as they do not comply with the basic rule. To solve this problem, we have conducted a thorough redefinition of the Left and Right sets of terminals that are the basis for calculating the precedence relations, and we have defined a new Leftmost set. The algorithms for calculating them are also described in detail. Our work's most significant contribution is that we establish precedence relationships between terminals by overcoming the basic rule of not having two consecutive non-terminals using an algorithm that allows building the operator precedence table for a CFG regardless of whether it is an OPG. The paper shows the complexities of the proposed algorithms and possible exceptions to the proposed rules. We present examples by using an OPG and two non-OPGs to illustrate the operation of the proposed algorithms. With these, the operator precedence table is built, and bottom-up parsing is carried out correctly.

Keywords: operator precedence language; operator precedence grammar; precedence table; precedence relations; algorithm



Citation: Lizcano, L.; Angulo, E.; Márquez, J. Precedence Table Construction Algorithm for CFGs Regardless of Being OPGs. *Algorithms* **2024**, *17*, 345. <https://doi.org/10.3390/a17080345>

Academic Editor: Dimitris Fotakis

Received: 24 June 2024

Revised: 29 July 2024

Accepted: 30 July 2024

Published: 7 August 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

This paper delves into the operator precedence languages family (OPL), created by R. Floyd [1], which is used for efficient bottom-up parsing [2,3]. The author's inspiration from the structure of arithmetic expressions, where precedence is assigned to multiplicative operators ($*$ and $/$) over additive operators ($+$ and $-$), is a key insight. Through the process of bottom-up parsing, the left side (non-terminal symbol) completely replaces the identified right-side production (body) in a context-free grammar (CFG), leaving no room for ambiguity. Operator precedence was extended to grammars that consist of terminals on the right side, even if non-terminals separated them. A CFG is generally considered an operator precedence grammar (OPG) if it does not contain productions with adjacent non-terminal symbols on its right-hand side.

OPLs have greatly aided in inferring context-free grammars. An example is described in [4–6], where a method is presented for inferring operator precedence-free grammars from bracketed samples. The method is based on constructing parse trees and utilizing functions that produce the grammars. Precedence rules are used to identify the brackets surrounding the terminal elements of a string in the sample, indicating the order in which the string should be evaluated.

Research on input-driven formal languages (IDL) [7–9] (later renamed as Visibly Pushdown Languages (VPL) [10]) has concluded [11] that OP closure properties imply ID closure properties and that ID languages are a specific type of OP languages characterized by limited OP relations [12].

Since the advancements in parallel computing, local parsability properties derived from OPL have been utilized to generate fast parallel parsers [13].

This paper's main contribution is its novel approach to building precedence relationships between terminal symbols in non-OPGs. This approach allows the generation of operator precedence tables and bottom-up parsing on a string derived by the non-OPG with support from the table obtained. The novelty of the work is a significant aspect that sets it apart from previous research in the field.

- The sets of Left and Right terminals proposed in previous works [1,11,14] to build precedence relationships between the terminals of an OPG were redefined.
- A new set of terminals called Leftmost was defined to support solving precedence relationships when there are two or more adjacent terminals on the right side of the productions. This set is significant as it systematically handles the precedence relationships in such complex scenarios, expanding the bottom-up parsing process with non-OPGs.
- An algorithm was established to build the precedence table with the support of the three sets of terminals proposed.
- Finally, applying the proposed algorithms makes it possible to obtain from a non-OPG the relations and operator precedence table previously limited to OPGs only. In addition, a bottom-up operator-precedence parser could be used to parse any string generated by a non-OPG, a technique that could only be applied to OPGs.

As demonstrated in this work, the proposed solution to construct the operator precedence table is not just a theoretical concept but a practical tool that can be applied to OPGs without any restrictions. At the end of the work, the functionality is demonstrated by applying it to an OPG and two non-OPGs and carrying out the respective bottom-up parsing. This practical demonstration should instill confidence in the effectiveness of the proposed solution and ensure its practical applicability.

The paper is organized as follows: Section 2 provides the necessary definitions to understand this work. Section 3 presents the previous work carried out in the area of CFG. Section 4 introduces our approach, detailing the problem we aim to solve and the redefinitions of previous works' concepts used to develop our algorithms. Section 5 presents examples of applying the proposed algorithm to one OPG and two non-OPGs, demonstrating the application of bottom-up parsing with the precedence tables to test their use. Section 6 describes several exceptions where the rules described in the proposed algorithms do not apply. Finally, Section 7 provides conclusions to our work.

2. Basic Definitions

2.1. Context-Free Grammars and Languages

Within Chomsky's hierarchy [15], context-free grammars play a significant role in programming and compilation language applications by addressing the syntactic structure of programming languages.

A *context-free grammar* (CFG) G generates a language $L(G)$, commonly called *context-free language* (CFL), which is composed of the strings generated by the CFG. The CFG is defined as a 4-tuple $G(S) = \langle T, N, S, P \rangle$, where:

- T represents the set of terminal symbols that create the strings within the CFL generated by the CFG.
- N represents the set of non-terminal symbols or syntactic variables that determine strings leading to CFL generation.
- S is the initial non-terminal symbol from which all strings defined by the CFG are generated.
- P is the set of productions or rules that group terminals and non-terminals to generate the strings. A production takes the form $A \rightarrow \alpha$, where A represents a non-terminal and α is a combination of terminals and non-terminals. The $\rightarrow$ is pronounced "produce". Thus, the production is read " A produce α ".

The following notation will be used: lowercase letters from the beginning of the alphabet represent terminal symbols ($a, b, c, \dots \in T$); uppercase letters from the beginning of the alphabet represent non-terminal symbols ($A, B, C, \dots \in N$); lowercase letters late in the alphabet represent terminal strings ($u, v, w, x, y, z \in T^+$); lowercase Greek letters generally represent terminal and/or non-terminal strings ($\alpha, \beta, \gamma, \dots \in (T \cup N)^*$), and the empty string will be represented by ε (this Greek letter will be the exception to the above convention).

2.2. Operator Precedence Grammars (Basic Rule)

A CFG is considered an operator grammar if none of the productions in P have adjacent non-terminal symbols on their right side. In other words, for a production $A \rightarrow \alpha$ in G , α does not take the form $uBCv$.

2.3. Derivations

A string $\eta \in (T \cup N)^*$ can be derived in k steps from a non-terminal $A \in N$ if there exists a sequence of strings of the form $\eta_1, \eta_2, \dots, \eta_k$. Using the symbol $\Rightarrow$, meaning *one-step derivation*, the sequence of strings is arranged in the form: $\eta_1 \Rightarrow \eta_2 \Rightarrow \dots \Rightarrow \eta_k$.

In this sequence, η_1 corresponds to A and η_k to η . Each intermediate η_i in the sequence has the form $\delta B \gamma$, where $B \in N$ and there is a production $B \rightarrow \beta$, such that by substituting it on the right side, $\eta_{i+1} (= \delta \beta \gamma)$ is obtained. The *derivation in one or more steps* is denoted with the operator $\Rightarrow^+$. In general, it is stated that every string of terminal symbols $w \in L(G)$ can be established as $S \Rightarrow^+ w$ or expanding the derivation in one or more steps: $S = \eta_1 \Rightarrow \eta_2 \Rightarrow \dots \Rightarrow \eta_k = w$. For a grammar G , a sentential form $\eta \in (T \cup N)^*$ is a string that $S \xRightarrow{*} \eta$, meaning that η can be obtained with zero or more derivations from S . If two grammars generate identical languages, they are considered *equivalent*.

2.4. Types of Derivations

According to [16], $S = \eta_1 \Rightarrow \eta_2 \Rightarrow \dots \Rightarrow \eta_k = w$ is a leftmost derivation of the string $w \in T^*$, when each $\eta_i, 2 \leq i \leq k-1$, has the form $x_i A_i \beta_i$ with $x_i \in T^*$, $A_i \in N$ and $\beta_i \in (T \cup N)^*$. Furthermore, $A_i \rightarrow \alpha_i$ belongs to P , and then, in each η_i , the symbol A_i is substituted by α_i , resulting in the sentential form η_{i+1} . Each η_i is called the left-sentential form.

If each $\eta_i, 2 \leq i \leq k-1$, takes the form of $\beta_i A_i x_i$ with $x_i \in T^*$, $A_i \in N$, and $\beta_i \in (T \cup N)^*$, then a rightmost derivation would be formed. Each η_i is called the right-sentential form.

2.5. Parse Trees

According to reference [16], a derivation tree for a CFG $G(S) = \langle T, N, S, P \rangle$ is a labeled and ordered tree in which each node receives a label corresponding to a symbol from the set $N \cup T \cup \{\varepsilon\}$. If a non-leaf node is labeled A and its immediate descendants are labeled $X_1, X_2, \dots, X_n$, then $A \rightarrow X_1 X_2 \dots X_n$ is a production in P . A labeled ordered tree D is a derivation tree for a CFG $G(A) = \langle T, N, A, P \rangle$ if

1. The root of D is labeled A .
2. If $D_1, \dots, D_k$ are the subtrees of the direct descendants of the root and the root of D_i is labeled X_i , then $A \rightarrow X_1 \dots X_k$ is in P .
3. D_i is a derivation tree for $G(X_i) = \langle T, N, X_i, P \rangle$ if X_i is in N or D_i is a single node labeled X_i if X_i is in T or D_i is a single node labeled ε .

Parse trees can be constructed from any derivation type.

2.6. Bottom-Up Parsing

The bottom-up parsing syntax analysis [2], also known as shift-reduce parsing, attempts to build a parsing tree for an input string that starts in the leaves (tree bottom) and moves to the root (tree top). This process is considered as the reduction of the string w to the initial symbol S of a CFG. At each step of parsing reduction, a particular substring of the sentential form that matches the right side of a production is replaced by the non-terminal symbol of the left side of that production. If, at each step, the substring is chosen correctly, a rightmost derivation is traced out inversely.

2.7. Handle

A handle [2,17] of a right-sentential form γ is a production $A \rightarrow \beta$ and a position of γ where the string β could be found and replaced by A to produce the previous right-sentential form in a rightmost derivation of γ . Generally, if $S \xRightarrow{*} \alpha Aw \Rightarrow \alpha \beta w$, then $A \rightarrow \beta$ if the position next to α is a handle of $\alpha \beta w$. If the grammar is ambiguous, more than one handle will be obtained. If the grammar is unambiguous, then every right-sentential form has exactly one handle. Usually, the string β is told to be a handle of $\alpha \beta w$ when the conditions for doing so are clear.

2.8. Implementation of a Bottom-Up Parsing

The bottom-up parsing by shift and reduction uses a stack to store grammar symbols and a buffer to handle the input string to be analyzed. The $\$$ symbol is also used to delimit the bottom of the stack and the right side of the input buffer. The recognition model format starts with the stack at $\$$ and the string $w\$$ in the input, as shown in Table 1.

Table 1. Bottom-up parsing table.

Stack	Input	Action
$\$$	$w\$$	
$\dots$	$\dots$	Shift/Reduce
$\dots$	$\dots$	Shift/Reduce
$\$S$	$\$$	<i>Accept</i>

The bottom-up parsing shifts zero or more input symbols to the stack until a handle is at the top. The parsing then reduces the handle to the left side of the appropriate production to obtain the sentential form corresponding to the previous step of the rightmost derivation. When reducing, the parsing recognizes that the right end of the handle is at the top of the stack. Therefore, the parsing takes action to find its left end in the stack and determine the non-terminal that will replace it according to the right side of some production where it matches the handle. These steps are repeated until an error is detected or until the stack contains $\$S$ and the input is with the symbol $\$$, at which point the parsing ends and the input string is considered valid for the CFG.

3. Previous Work

According to [1,11,14], for a CFG G , the left terminal sets $\mathcal{L}_G(A)$ and right terminal sets $\mathcal{R}_G(A)$ are defined as follows:

$$\mathcal{L}_G(A) = \{a | A \xRightarrow{*} \gamma a \alpha, \gamma \in N \cup \{\varepsilon\}\}.$$

$$\mathcal{R}_G(A) = \{a | A \xRightarrow{*} \alpha a \gamma, \gamma \in N \cup \{\varepsilon\}\}.$$

Precedence relationships arise from the parse tree's establishment of binary relationships between consecutive terminals or those that become sequential after a bottom-up process toward the non-terminal symbol S .

The precedence relationships are established as follows:

1. $a \doteq b$, if and only if $\exists A \rightarrow \alpha a \gamma b \beta \in P, \gamma \in N \cup \{\varepsilon\}$.
2. $a < b$, if and only if $\exists A \rightarrow \alpha a D \beta \in P, b \in \mathcal{L}_G(D)$.
3. $a > b$, if and only if $\exists A \rightarrow \alpha D b \beta \in P, a \in \mathcal{R}_G(D)$.

According to [2], precedence relations are used to delimit a handle in a right-sentential form. That is, in the sentential form $\alpha\beta w$, the substring β is a handle if there is a production $A \rightarrow \beta$ in the CFG; therefore, it can be delimited by the precedence relations, obtaining $\alpha < \beta > w$. The relation $\doteq$ between the symbols used when the handle has more than one consecutive terminal symbol. That is, if $\beta = a_1 a_2 \dots a_n$, then the relation $a_i \doteq a_{i+1}, \forall i = 1, \dots, n-1$ should be established. Suppose a right-sentential form is $A_0 a_1 A_1 a_2 \dots a_n A_n$, where each A_i is a non-terminal. In that case, the relations can be established as $a_i < A_i a_{i+1}$, with a maximum of one non-terminal (operator grammar principle) between the terminals. Non-terminals could be eliminated in the right sentential form, leaving only the relationships between terminals.

In short, each time a handle is obtained, it can be enclosed between the symbols of $<$ and $>$. In addition, two consecutive terminal symbols can be inside a handle (non-terminals in the middle), and the relationship is to be established $\doteq$. Finally, non-terminal symbols can be removed from the right-sentential form when a handle is found and locked between the precedence relations.

According to [2], the concepts discussed in Section 2.8, and the precedence relations, the precedence parsing Algorithm 1 is constructed as follows:

Algorithm 1 Precedence parsing algorithm

Require: Set of precedence relations.

```

procedure PRECEDENCEPARSING( $w$ )
     $stack \leftarrow [\$]$ 
     $end \leftarrow false$ 
     $pos \leftarrow 0$ 
    do
         $a \leftarrow stack[top]$ 
         $b \leftarrow w[pos]$ 
        if  $a = \$$  and  $b = \$$  then
            print("Action: Accept")
             $end \leftarrow true$ 
        else
            if  $a < b$  or  $a \doteq b$  then
                 $pos++$ 
                push( $stack, b$ )
                print("Action: Shift")
            else
                if  $a > b$  then
                    do
                         $e \leftarrow pull(stack)$ 
                        while not ( $stack[top] < e$ )
                            print("Action: Reduce")
                    else
                        print("Action: Error")
                         $end \leftarrow true$ 
                    end if
                end if
            end if
        while not  $end$ 
    end procedure

```

4. Approach

This work proposes a redefinition of the *Left* and *Right* terminal sets, corresponding to $\mathcal{L}_G(A)$ and $\mathcal{R}_G(A)$, respectively. Additionally, the definition of the new *Leftmost* set is presented. The new rules for determining precedence relations are demonstrated even if the CFG does not comply with the fundamental rule of operator grammar, meaning that the CFG can have more than one adjacent non-terminal symbol on the right side of its productions.

We present the algorithms for constructing the three proposed sets: *Left*, *Leftmost*, and *Right*. Additionally, we demonstrate the algorithm that incorporates the newly established precedence rules.

4.1. Definitions

There are the following symbols:

- A string of terminals $x \in T^+$.
- A string of non-terminals $\beta \in N^*$.
- A sentential form $\gamma \in (T \cup N)^*$.

The following sets are defined:

- The left, denoted as $\mathcal{L}(A)$, is the set of terminal symbols that can appear on the left side of any sentential form as a product of a rightmost derivation from A .

$$\mathcal{L}(A) = \{a \in T \mid A \xRightarrow{*} \beta a \gamma\}.$$

- The leftmost, denoted as $\mathcal{L}_m(A)$, is the set of terminal symbols that can appear at the end of the left side in any derivation from A , in the absence of other leftmost grammatical symbols (or that derive to ϵ in their absence).

$$\mathcal{L}_m(A) = \{a \in T \mid A \xRightarrow{*} a \gamma\}.$$

- The right, denoted as $\mathcal{R}(A)$, is the set of terminal symbols that can appear at the right end of the rightmost derivations from A or appear further to the right in any production of A .

$$\mathcal{R}(A) = \{a \in T \mid A \xRightarrow{*} \gamma a \vee A \rightarrow \gamma a \beta\}.$$

According to the above, the following precedence rules are defined:

- If $C \rightarrow \gamma_1 a \beta B \gamma_2$, $b \in \mathcal{L}(B)$, then $a < b$.
- If $C \rightarrow \gamma_1 A \beta B \gamma_2$, $a \in \mathcal{R}(A) \wedge b \in \mathcal{L}_m(B) \wedge \beta \xRightarrow{*} \epsilon$, then $a > b$.
- If $C \rightarrow \gamma_1 A \beta b \gamma_2$, $a \in \mathcal{R}(A) \wedge \beta \xRightarrow{*} \epsilon$, then $a > b$.
- If $C \rightarrow \gamma_1 a \beta b \gamma_2$, then $a \doteq b$.

4.2. Problem Analysis

4.2.1. Absolute Operator Grammar

Given the rightmost derivation series of the form:

$$S \xRightarrow{+} \gamma_1 A x_1 \xRightarrow{*} \gamma_1 \omega_1 x_1, \text{ where } A \rightarrow \omega_1 \text{ and } \omega_1 = \gamma_2 B \gamma_3. \quad (1)$$

It is noted that ω_1 is a handle according to the previous definition in Section 2.7.

The right-sentential form of (1) can be rewritten as follows:

$$\gamma_1 \omega_1 x_1 = \gamma_1 \gamma_2 B \gamma_3 x_1 \xRightarrow{+} \gamma_1 \gamma_2 B x_2 x_1 \xRightarrow{*} \gamma_1 \gamma_2 \omega_2 x_2 x_1, \text{ where } B \rightarrow \omega_2. \quad (2)$$

From the last sentential form in (2), it follows that ω_2 is a handle according to the definition in Section 2.7.

Because of the above, the analysis will focus on the productions born of A , which have at least one non-terminal B on the right, as seen in (1). The productions in A will be traversed through grammar to find the relations of precedence of non-terminal B located on the right side.

Within the context of ω_1 , having to $B \rightarrow \omega_2$, and ω_2 is a handle, the following forms can be obtained from its definition in (1):

- (a) $\omega_1 = \gamma_4 a B$, it follows that $a < \mathcal{L}(B)$. Here $\gamma_2 = \gamma_4 a$ and $\gamma_3 = \varepsilon$.
- (b) $\omega_1 = B b \gamma_5$, it follows that $\mathcal{R}(B) > b$. Here $\gamma_2 = \varepsilon$ and $\gamma_3 = b \gamma_5$.
- (c) $\omega_1 = \gamma_4 a B b \gamma_5$, it follows that $a < \mathcal{L}(B)$ and $\mathcal{R}(B) > b$. Here $\gamma_2 = \gamma_4 a$ and $\gamma_3 = b \gamma_5$.

For the non-terminal start symbol, S , an initial dummy production is defined $E \rightarrow \$S\$$ and rule (c) is applied.

For a production $A \rightarrow \gamma_1 a X b \gamma_2$, $X \in N \cup \{\varepsilon\}$, it follows that $a \doteq b$.

4.2.2. General Grammar with Two or More Consecutive Non-Terminal Symbols

Redefinition of Left, Right and definition of Leftmost.

Left(B):

Given the rightmost derivation:

$$\gamma_1 a B x_1 \Rightarrow \gamma_1 a \omega_1 x_1. \quad (3)$$

It is observed that ω_1 is a handle within a bottom-up parsing (inverse rightmost derivation) since $B \Rightarrow \omega_1$; therefore, from (3), it can be established that $\gamma_1 a < \omega_1 > x_1$. In addition, ω_1 can have the following forms:

- (a) $\omega_1 = \beta_1 b \gamma_2$.
- (b) $\omega_1 = \beta_1 C \gamma_3$.

Continuing with the rightmost derivation in (3) and taking each of the options of ω_1 , we obtain that:

- (a) $\gamma_1 a B x_1 \Rightarrow \gamma_1 a \beta_1 b \gamma_2 x_1$ since ω_1 is a handle, consequently $\gamma_1 a < \beta_1 b \gamma_2 > x_1$; therefore, $a < b$ where $B \rightarrow \beta_1 b \gamma_2$ and $b \in \mathcal{L}(B)$.
- (b) $\gamma_1 a B x_1 \Rightarrow \gamma_1 a \beta_1 C \gamma_3 x_1 \xRightarrow{*} \gamma_1 a \beta_1 C x_2 x_1 \xRightarrow{*} \gamma_1 a \beta_1 \beta_2 D x_3 x_2 x_1 \Rightarrow \gamma_1 a \beta_1 \beta_2 \beta_3 b \gamma_4 x_3 x_2 x_1$. It is observed that $C \xRightarrow{+} \beta_2 D x_3$ and $D \rightarrow \beta_3 b \gamma_4$; therefore, $\beta_3 b \gamma_4$ is a handle, and consequently $\gamma_1 a \beta_1 \beta_2 < \beta_3 b \gamma_4 > x_3 x_2 x_1$ and $a < b$.

In short, $\gamma_1 a B x_1 \xRightarrow{+} \gamma_1 a \beta_1 \beta_2 \beta_3 b \gamma_4 x_3 x_2 x_1$; taking $\beta = \beta_1 \beta_2 \beta_3$ and $\gamma_5 = \gamma_4 x_3 x_2$, we obtain the summary derivation $\gamma_1 a B x_1 \xRightarrow{+} \gamma_1 a \beta b \gamma_5 x_1$, where $B \xRightarrow{+} \beta b \gamma_5$ and $b \in \mathcal{L}(B)$.

Right(B):

Given the rightmost derivation:

$$\gamma_1 B a x_1 \Rightarrow \gamma_1 \omega_1 a x_1. \quad (4)$$

It is observed that ω_1 is a handle within a bottom-up parsing (inverse rightmost derivation), since $B \Rightarrow \omega_1$; therefore, from (4), it can be established that $\gamma_1 < \omega_1 > a x_1$. In addition, ω_1 can have the following forms:

- (a) $\omega_1 = \gamma_2 b \beta$.
- (b) $\omega_1 = \gamma_3 C \beta_1, \beta_1 \xRightarrow{*} \varepsilon$.

Continuing with the rightmost derivation in (4) and taking each of the options of ω_1 , we obtain that:

- (a) $\gamma_1 B a x_1 \Rightarrow \gamma_1 \gamma_2 b \beta a x_1$ since ω_1 is a handle, consequently $\gamma_1 < \gamma_2 b \beta > a x_1$; therefore, $b > a$ where $B \rightarrow \gamma_2 b \beta$ and $b \in \mathcal{R}(B)$.
- (b) $\gamma_1 B a x_1 \Rightarrow \gamma_1 \gamma_3 C \beta_1 a x_1 \xRightarrow{*} \gamma_1 \gamma_3 \gamma_4 D a x_1 \Rightarrow \gamma_1 \gamma_3 \gamma_4 \gamma_5 b \beta_2 a x_1$. It is noted that $C \xRightarrow{+} \gamma_4 D$ and $D \rightarrow \gamma_5 b \beta_2$; therefore, $\gamma_5 b \beta_2$ is a handle, and consequently,

$$C \longrightarrow cD$$

$$D \longrightarrow d$$

$$F \longrightarrow f$$

The rightmost derivation of the string $abefcd$ carries the following way of creating the $Right(A)$ or $\mathcal{R}(A)$.

$$A \Longrightarrow \underset{b}{a\mathbf{b}}BC \Longrightarrow ab\underset{c}{Bc}D \Longrightarrow abB\underset{d}{c\mathbf{d}} \Longrightarrow abeFcd \Longrightarrow abefcd. >$$

The terminal symbols b , c , and d are viewed from the right and recorded in $\mathcal{R}(A)$. The symbol b is the first one observed and recorded in $\mathcal{R}(A)$. The symbol a is blocked by b , since both appear when A is derived. C is then replaced by cD and c is observed and recorded in $\mathcal{R}(A)$. When D is substituted for d , we observe and record d in $\mathcal{R}(A)$. Then B is derived by eF but e cannot be recorded since the previous inclusion of c and d in $\mathcal{R}(A)$ makes it invisible or blocks it from being observed from the right. Finally, when substituting F for f , it cannot be observed from the right either (blocked by c and d) and cannot be recorded. Consequently:

$$\mathcal{R}(A) = \{b, c, d\}.$$

Vision of Leftmost(.):

Given the CFG:

$$S \longrightarrow dAB$$

$$A \longrightarrow fa$$

$$B \longrightarrow Cb$$

$$C \longrightarrow c$$

The rightmost derivation of the string $dfacb$ carries the following way of creating the $Leftmost(B)$ or $\mathcal{L}_m(B)$, as shown in Figure 1.

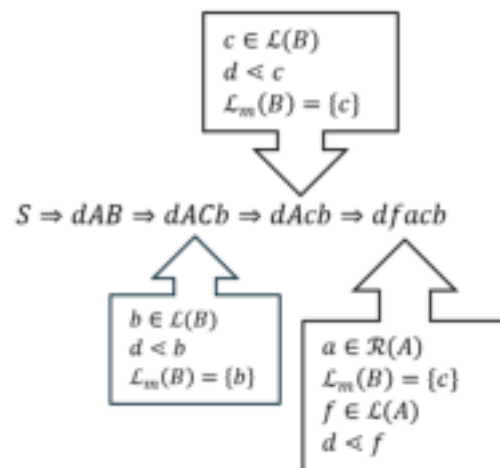


Figure 1. Derivation of $dfacb$ chain and $Leftmost(B)$ set creation sequence.

It can be seen that $\mathcal{L}(B) = \{b, c\}$. When B derivates Cb , we observe that $\mathcal{L}_m(B) = \{b\}$ because no more terminal symbols are seen up to that point. In the following derivation, C is replaced by c ; therefore, c is located further to the left of the substring cb , leaving $\mathcal{L}_m(B) = \{c\}$.

4.2.4. Summary of Relations between Grammatical Symbols from Productions and Sets

Table 2 shows the relationships between the grammatical symbols of the productions and the sets of terminals.

Table 2. General productions and precedence relationships obtained.

Producing	Relationship
$A \rightarrow a\beta b$	$a \doteq b$
$A \rightarrow C\beta b, \beta \xRightarrow{*} \varepsilon$	$\mathcal{R}(C) \succ b$
$A \rightarrow a\beta C$	$a \prec \mathcal{L}(C)$
$A \rightarrow B\beta C, \beta \xRightarrow{*} \varepsilon$	$\mathcal{R}(B) \succ \mathcal{L}_m(C)$

4.3. Algorithms

The following algorithms were developed based on the definitions presented in Section 4.1.

Algorithm 2 receives a non-terminal A whose productions are known; for each of these, it is verified whether they correspond to the form $\beta B\gamma$; if so, Algorithm 2 is executed again, sending the non-terminal B as a parameter (provided that $B \neq A$), and its response is included in the set $\mathcal{L}(A)$.

Subsequently, it is verified whether the production corresponds to the form $\beta a\gamma$, and if so, the terminal a is added to the set $\mathcal{L}(A)$.

Once the verification of all productions of A is completed, the set $\mathcal{L}(A)$ is returned as the output of Algorithm 2.

Algorithm 2 Left algorithm of a non-terminal A

```

function LEFT( $A$ )
   $\mathcal{L}(A) \leftarrow \emptyset$ 
  for each production of  $A$  do
    if  $A \rightarrow \beta B\gamma$  then
       $\mathcal{L}(A) \supseteq \text{Left}(B)$  //  $\mathcal{L}(A)$  contains a Left( $B$ )
    end if
    if  $A \rightarrow \beta a\gamma$  then
       $a \in \mathcal{L}(A)$  //  $a$  is in  $\mathcal{L}(A)$ 
    end if
  end for
  return  $\mathcal{L}(A)$ 
end function

```

Algorithm 3 receives a non-terminal A whose productions are known; for each of these, it is verified if they correspond to the form $\beta B\gamma$ and $\beta \xRightarrow{*} \varepsilon$; if so, Algorithm 3 is executed again, sending the non-terminal B as a parameter (provided that $B \neq A$), and its response is included in the set $\mathcal{L}_m(A)$.

Subsequently, it is verified if the production corresponds to the form $\beta a\gamma$ and $\beta \xRightarrow{*} \varepsilon$. If so, the terminal a is added to the set $\mathcal{L}_m(A)$.

Once the verification of all productions of A is completed, the set $\mathcal{L}_m(A)$ is returned as the output of Algorithm 3.

Algorithm 3 Leftmost algorithm of a non-terminal A

```

function LEFTMOST( $A$ )
   $\mathcal{L}_m(A) \leftarrow \emptyset$ 
  for each production of  $A$  do
    if  $A \rightarrow \beta B \gamma$  and  $\beta \xRightarrow{*} \varepsilon$  then
       $\mathcal{L}_m(A) \supseteq \text{Leftmost}(B)$ 
    end if
    if  $A \rightarrow \beta a \gamma$  and  $\beta \xRightarrow{*} \varepsilon$  then
       $a \in \mathcal{L}_m(A)$ 
    end if
  end for
  return  $\mathcal{L}_m(A)$ 
end function

```

Algorithm 4 receives a non-terminal A whose productions are known; for each of these, it is verified if they correspond to the form $\gamma B \beta$ and $\beta \xRightarrow{*} \varepsilon$; if so, Algorithm 4 is executed again, sending the non-terminal B as a parameter (provided that $B \neq A$) and its response is included in the set $\mathcal{R}(A)$.

Subsequently, it is verified whether the production corresponds to the form $\gamma a \beta$, and if so, the terminal a is added to the set $\mathcal{R}(A)$.

Once all productions of A have been verified, the set $\mathcal{R}(A)$ is returned as the output of Algorithm 4.

Algorithm 4 Right algorithm of a non-terminal A

```

function RIGHT( $A$ )
   $\mathcal{R}(A) \leftarrow \emptyset$ 
  for each production of  $A$  do
    if  $A \rightarrow \gamma B \beta$  and  $\beta \xRightarrow{*} \varepsilon$  then
       $\mathcal{R}(A) \supseteq \text{Right}(B)$ 
    end if
    if  $A \rightarrow \gamma a \beta$  then
       $a \in \mathcal{R}(A)$ 
    end if
  end for
  return  $\mathcal{R}(A)$ 
end function

```

Algorithm 5 receives a grammar $G(S) = \langle T, N, S, P \rangle$ and Algorithms 2–4. Initially, precedence relations are added between the delimiting symbol $\$$ and the Left and Right sets of the non-terminal symbol of start S , according to rules (A.1) and (A.2), respectively.

Algorithm 5 operates in an iterative manner. It traverses all the productions of the set P , and for each iteration, it initializes two temporary variables, u and l , to null and an empty list, respectively. It then performs a series of processes for each pair of contiguous grammatical symbols X and Y in a production.

1. The algorithm checks if XY has the form ab ; if so, it adds the corresponding precedence relationship between a and b , following rule (B).
2. The algorithm checks if XY has the form aB ; if so, it assigns u the value of a and adds the corresponding precedence relations between a and terminals in $\text{Left}(B)$, following rule (C).
3. The algorithm checks if XY has the form Ab ; if so, it adds the corresponding precedence relations between b and terminals in $\text{Right}(A)$, following rule (D.1). Then, check if $u \neq \text{null}$, and if so, add the corresponding precedence relationship between u and b , following rule (D.2), and reset u to null. Finally, checks if list l has elements, and if so, adds the corresponding precedence relations between b and terminals in $\text{Right}(B)$, following rule (D.3), and resets list l to empty.

4. The algorithm checks if XY has the form AB ; if so, it adds the corresponding precedence relations between terminals in $\text{Right}(A)$ and terminals in $\text{Leftmost}(B)$, following rule (E.1). Then, it checks if $u \neq \text{null}$, and if so, adds the corresponding precedence relation between u and terminals in $\text{Left}(B)$, following rule (E.2). Finally, it checks if the list l has elements and if so, adds the corresponding precedence relations between terminals in $\text{Right}(C)$, with $C \in l$ and terminals in $\text{Leftmost}(B)$, following rule (E.3), and resets the list l if necessary.

Algorithm 5 Algorithm to obtain the operator precedence table of a G grammar

Require: $G(S) = \langle T, N, S, P \rangle$

```

procedure TABLE( $G, \text{Left}(\cdot), \text{Right}(\cdot), \text{Leftmost}(\cdot)$ )
  compute  $\text{Left}(A), \text{Right}(A), \text{Leftmost}(A), \forall A \in N$ 
  add  $\$ \prec a, \forall a \in \text{Left}(S)$  (A.1)
  add  $a \succ \$, \forall a \in \text{Right}(S)$  (A.2)
  for each  $A \rightarrow \gamma$  in  $P$  do
     $u \leftarrow \text{null}$ 
     $l \leftarrow []$  // empty list
    for each  $XY$  contiguous in  $\gamma$  do
      if  $XY = ab$  then
        add  $a \doteq b$  (B)
      end if
      if  $XY = aB$  then
         $u \leftarrow a$ 
        add  $a \prec b, \forall b \in \text{Left}(B)$  (C)
      end if
      if  $XY = Ab$  then
        add  $a \succ b, \forall a \in \text{Right}(A)$  (D.1)
        if  $u \neq \text{null}$  then
          add  $u \doteq b$  (D.2)
           $u \leftarrow \text{null}$ 
        end if
        if  $l \neq []$  then
          for  $B \in l$  do
            add  $a \succ b, \forall a \in \text{Right}(B)$  (D.3)
          end for
           $l \leftarrow []$ 
        end if
      end if
      if  $XY = AB$  then
        add  $a \succ b, \forall a \in \text{Right}(A) \wedge \forall b \in \text{Leftmost}(B)$  (E.1)
        if  $u \neq \text{null}$  then
          add  $u \prec b, \forall b \in \text{Left}(B)$  (E.2)
        end if
        if  $l \neq []$  then
          for  $C \in l$  do
            add  $c \succ b, \forall c \in \text{Right}(C) \wedge \forall b \in \text{Leftmost}(B)$  (E.3)
          end for
        end if
        if  $B \xRightarrow{*} \varepsilon$  then
          add  $A$  to  $l$ 
        else
           $l \leftarrow []$ 
        end if
      end if
    end for
  end for
end procedure

```

4.4. Complexity Analysis

The following variables will be established for the computation of the complexity of the four proposed algorithms:

- t : Cardinality of the set of terminal symbols.
- p : Cardinality of the set of productions.
- g : Number of grammatical symbols in the productions.
- n : Cardinality of the set of non-terminal symbols.

4.4.1. Complexity of Algorithm 2 (Left)

The **Left** function checks in which productions the symbol A appears on the left side. In the first **if-block**, a recursive call of **Left** is observed, which implies that all non-terminal symbols of G could be traversed at the end of the execution in the search for **Left**(A). Therefore, all worst-case productions would be traversed, yielding a computational time of $\mathcal{O}(p)$. The second **if-block** has a computational time of $\mathcal{O}(1)$, which does not contribute to the overall computational time $\mathcal{O}(p)$.

4.4.2. Complexity of Algorithm 3 (Leftmost)

The **Leftmost** function has a similar structure to the previous one; therefore, the computational execution time is $\mathcal{O}(p)$.

4.4.3. Complexity of Algorithm 4 (Right)

The **Right** function has a similar structure to **Left**; therefore, the computational execution time is $\mathcal{O}(p)$.

4.4.4. Complexity of Algorithm 5 (Table)

According to the first **for-cycle**, the number of times it would be executed will be p . Immediately following the previous one, the inner **for-cycle** is executed in the worst-case g times, assuming that the total number of grammatical symbols appear in each production. Therefore, the computational cost of the two cycles is $\mathcal{O}(p \times g)$.

The computational costs for the construction of **Left**(A), **Right**(A), and **Leftmost**(A), $\forall A \in N$, are already computed. In computing the complexity of **Table**, the length of each generated set will be considered, which in the worst case is t .

The analysis of the four **if-blocks** inside the second **for-cycle** is as follows:

- In the first **if-block**, after checking, the executed instruction is carried out in constant time and is denoted $\mathcal{O}(1)$.
- In the second **if-block**, after checking, the first instruction is executed in a constant time $\mathcal{O}(1)$, and the second is executed in a computational time $\mathcal{O}(t)$; therefore, the computational time of the **if-block** is $\mathcal{O}(t)$.
- In the third **if-block**, after checking, three sub-blocks are distinguished; the first one corresponds to the travel of **Right**(A), executed in a computational time $\mathcal{O}(t)$. After checking $u \neq null$, the second sub-block presents two instructions executed in a constant time $\mathcal{O}(1)$. The third sub-block, after the checking of $l \neq []$, presents a computational cost $\mathcal{O}(n \times t)$, determined by the **for-cycle** executed, in the worst case, $\mathcal{O}(n)$ times, corresponding to the number of non-terminal symbols in G and the travel it makes from **Right**(B) with a cost $\mathcal{O}(t)$. In summary, the computational cost of the third **if-block** is $\mathcal{O}(n \times t)$, corresponding to the worst case.
- In the fourth **if-block**, after checking, four sub-blocks are distinguished. The first corresponds to the combinations formed from traversing **Right**(A) and **Leftmost**(B) to obtain the precedence relations $a \succ b$. Since the maximum size of each set is t , the computational cost of these combinations is $\mathcal{O}(t^2)$. After checking $u \neq null$, the second sub-block presents an instruction where **Left**(B) is traversed; therefore, the computational cost is $\mathcal{O}(t)$. In the third sub-block, after the checking of $l \neq []$, it presents a **for-cycle** that executes at most n times the combinations formed from

traversing $\text{Right}(C)$ and $\text{Leftmost}(B)$ to obtain the precedence relations $c \succ b$. These combinations are traversed maximally in $\mathcal{O}(t^2)$ computational time because t is the size of both sets and, consequently, the computational cost of the third sub-block is $\mathcal{O}(n \times t^2)$. The last sub-block will have a cost of $\mathcal{O}(1)$. In summary, the computational cost of the fourth block is $\mathcal{O}(n \times t^2)$, corresponding to the worst case.

In summary, the computational cost of the four **if-blocks** within the second **for-cycle** of Algorithm 5 is $\mathcal{O}(n \times t^2)$, this being the worst case; therefore, the computational cost of the main body of Algorithm 5 formed by the two nested **for-cycles** and the four **if-blocks** is $\mathcal{O}(p \times g \times n \times t^2)$ which is greater than the cost of computing any of the three sets of terminal symbols $\text{Left}(A)$, $\text{Right}(A)$, and $\text{Leftmost}(A)$, $\forall A \in N$, when starting the algorithm, and corresponding to $\mathcal{O}(p)$.

5. Examples

5.1. Example 1

5.1.1. Grammar

Given the following OPG:

$$E \longrightarrow E + T \mid T$$

$$T \longrightarrow T * F \mid F$$

$$F \longrightarrow (E) \mid id$$

The sets are built step by step following the algorithms.

5.1.2. Left

The Left sets are calculated as detailed in Table 3.

Table 3. Left Sets calculation for the CFG in Example 1.

$\text{Left}(E)$			
For $E \longrightarrow E + T$			
1. Evaluating the first conditional of Algorithm 2.	$A = E$ $\beta = \varepsilon$ $B = E$ $\gamma = +T$ $\mathcal{L}(E) \supseteq \mathcal{L}(E)$	2. Evaluating the second conditional of Algorithm 2.	$A = E$ $\beta = E$ $a = +$ $\gamma = T$ $\mathcal{L}(E) \supseteq \{+\}$
For $E \longrightarrow T$			
3. Evaluating the first conditional of Algorithm 2.	$A = E$ $\beta = \varepsilon$ $B = T$ $\gamma = \varepsilon$ $\mathcal{L}(E) = \{+\} \cup \mathcal{L}(T)$		
$\text{Left}(T)$			
For $T \longrightarrow T * F$			
1. Evaluating the first conditional of Algorithm 2.	$A = T$ $\beta = \varepsilon$ $B = T$ $\gamma = *F$ $\mathcal{L}(T) \supseteq \mathcal{L}(T)$	2. Evaluating the second conditional of Algorithm 2.	$A = T$ $\beta = T$ $a = *$ $\gamma = F$ $\mathcal{L}(T) \supseteq \{*\}$
For $T \longrightarrow F$			
3. Evaluating the first conditional of Algorithm 2.	$A = T$ $\beta = \varepsilon$ $B = F$ $\gamma = \varepsilon$ $\mathcal{L}(T) = \{*\} \cup \mathcal{L}(F)$		

Table 3. Cont.

Left(F)	
For $F \rightarrow (E)$	For $F \rightarrow id$
1. Evaluating the second conditional of Algorithm 2. $A = F$ $\beta = \varepsilon$ $a = ($ $\gamma = E)$ $\mathcal{L}(F) \supseteq \{(\}$	2. Evaluating the second conditional of Algorithm 2. $A = F$ $\beta = \varepsilon$ $a = id$ $\gamma = \varepsilon$ $\mathcal{L}(F) = \{(\, id\}$

Summary of Left sets of non-terminals:

- Left(E) = $\{+\} \cup \mathcal{L}(T)$.
- Left(T) = $\{*\} \cup \mathcal{L}(F)$.
- Left(F) = $\{(\, id\}$.

Substituting the sets:

- Left(E) = $\{+, *, (\, id\}$.
- Left(T) = $\{*, (\, id\}$.
- Left(F) = $\{(\, id\}$.

5.1.3. Right

The Right sets are calculated as detailed in Table 4.

Table 4. Right Sets calculation for the CFG in Example 1.

Right(E)	
For $E \rightarrow E + T$	
1. Evaluating the first conditional of Algorithm 4. $A = E$ $\gamma = E +$ $B = T$ $\beta = \varepsilon$ $\mathcal{R}(E) \supseteq \mathcal{R}(T)$	2. Evaluating the second conditional of Algorithm 4. $A = E$ $\gamma = E$ $a = +$ $\beta = T$ $\mathcal{R}(E) \supseteq \mathcal{R}(T) \cup \{+\}$
For $E \rightarrow T$	
3. Evaluating the first conditional of Algorithm 4. $A = E$ $\gamma = \varepsilon$ $B = T$ $\beta = \varepsilon$ $\mathcal{R}(E) = \mathcal{R}(T) \cup \{+\}$	
Right(T)	
For $T \rightarrow T * F$	
1. Evaluating the first conditional of Algorithm 4. $A = T$ $\gamma = T *$ $B = F$ $\beta = \varepsilon$ $\mathcal{R}(T) \supseteq \mathcal{R}(F)$	2. Evaluating the second conditional of Algorithm 4. $A = T$ $\gamma = T$ $a = *$ $\beta = F$ $\mathcal{R}(T) \supseteq \mathcal{R}(F) \cup \{*\}$
For $T \rightarrow F$	
3. Evaluating the first conditional of Algorithm 4. $A = T$ $\gamma = \varepsilon$ $B = F$ $\beta = \varepsilon$ $\mathcal{R}(T) = \mathcal{R}(F) \cup \{*\}$	

Table 4. Cont.

Right(F)			
For $F \rightarrow (E)$		For $F \rightarrow id$	
1. Evaluating the second conditional of Algorithm 2.	$A = F$ $\gamma = (E$ $a =)$ $\beta = \varepsilon$ $\mathcal{R}(F) \supseteq \{()\}$	2. Evaluating the second conditional of Algorithm 2.	$A = F$ $\gamma = \varepsilon$ $a = id$ $\beta = \varepsilon$ $\mathcal{R}(F) = \{(), id\}$

Summary of Right sets of non-terminals:

- $\text{Right}(E) = \mathcal{R}(T) \cup \{+\}$.
- $\text{Right}(T) = \mathcal{R}(F) \cup \{*\}$.
- $\text{Right}(F) = \{(), id\}$.

Substituting the sets:

- $\text{Right}(E) = \{(), id, *, +\}$.
- $\text{Right}(T) = \{(), id, *\}$.
- $\text{Right}(F) = \{(), id\}$.

5.1.4. Leftmost

The Leftmost are calculated as detailed in Table 5.

Table 5. Leftmost Sets calculation for the CFG in Example 1.

Leftmost(E)			
For $E \rightarrow E + T$		For $E \rightarrow T$	
1. Evaluating the first conditional of Algorithm 3.	$A = E$ $\beta = \varepsilon$ $B = E$ $\gamma = +T$ $\mathcal{L}_m(E) \supseteq \mathcal{L}_m(E)$	2. Evaluating the first conditional of Algorithm 3.	$A = E$ $\beta = \varepsilon$ $B = T$ $\gamma = \varepsilon$ $\mathcal{L}_m(E) = \mathcal{L}_m(T)$

Leftmost(T)			
For $T \rightarrow T * F$		For $T \rightarrow F$	
1. Evaluating the first conditional of Algorithm 3.	$A = T$ $\beta = \varepsilon$ $B = T$ $\gamma = *F$ $\mathcal{L}_m(T) \supseteq \mathcal{L}_m(T)$	2. Evaluating the first conditional of Algorithm 3.	$A = T$ $\beta = \varepsilon$ $B = F$ $\gamma = \varepsilon$ $\mathcal{L}_m(T) = \mathcal{L}_m(F)$

Leftmost(F)			
For $F \rightarrow (E)$		For $F \rightarrow id$	
1. Evaluating the second conditional of Algorithm 3.	$A = F$ $\beta = \varepsilon$ $a = ($ $\gamma = E)$ $\mathcal{L}_m(F) \supseteq \{(\}$	2. Evaluating the second conditional of Algorithm 3.	$A = F$ $\beta = \varepsilon$ $a = id$ $\gamma = \varepsilon$ $\mathcal{L}_m(F) = \{(), id\}$

Summary of Leftmost sets of non-terminals:

- $\text{Leftmost}(E) = \mathcal{L}_m(T)$.

- $\text{Leftmost}(T) = \mathcal{L}_m(F)$.
- $\text{Leftmost}(F) = \{(\text{,id})\}$.

Substituting the sets:

- $\text{Leftmost}(E) = \{(\text{,id})\}$.
- $\text{Leftmost}(T) = \{(\text{,id})\}$.
- $\text{Leftmost}(F) = \{(\text{,id})\}$.

5.1.5. Case Studies

For Algorithm 5, the application of some precedence relations is exhibited.

- For the delimiter symbol \$:
 - $\$ \prec \mathcal{L}(E) \rightarrow \text{Rule (A.1)}.$
 - $\$ \prec +$
 - $\$ \prec *$
 - $\$ \prec ($
 - $\$ \prec id$
 - $\mathcal{R}(E) \succ \$ \rightarrow \text{Rule (A.2)}.$
 - $) \succ \$$
 - $id \succ \$$
 - $* \succ \$$
 - $+ \succ \$$
- For $E \rightarrow E + T$
 - $X = E, Y = + \rightarrow \text{Rule (D.1)}.$
 $\mathcal{R}(E) \succ +$
 - $) \succ +$
 - $id \succ +$
 - $* \succ +$
 - $+ \succ +$ $u = \text{null}, l = []$
 - $X = +, Y = T \rightarrow \text{Rule (C)}.$
 $+ \prec \mathcal{L}(T)$
 - $+ \prec *$
 - $+ \prec ($
 - $+ \prec id$ $u = +, l = []$
- For $F \rightarrow (E)$
 - $X = (, Y = E \rightarrow \text{Rule (C)}.$
 $(\prec \mathcal{L}(E)$
 - $(\prec +$
 - $(\prec *$
 - $(\prec ($
 - $(\prec id$ $u = (, l = []$
 - $X = E, Y =) \rightarrow \text{Rule (D.1)}.$
 $\mathcal{R}(E) \succ)$
 - $) \succ)$
 - $id \succ)$
 - $* \succ)$
 - $+ \succ)$ $u \neq \text{null} \rightarrow u \doteq b \rightarrow (\doteq) \rightarrow \text{Rule (D.2)}.$
 $u = \text{null}, l = []$

4. In the following cases, the form XY is not fulfilled in the right side part of the production; therefore, no rule can be applied:

- (a) $E \rightarrow T$
 (b) $T \rightarrow F$
 (c) $F \rightarrow id$

The remaining precedence relations are derived using the same method as the previously demonstrated cases.

5.1.6. Operator Precedence Table

At the end of the algorithm execution, operator precedence Table 6 is obtained.

Table 6. Precedence Table of Example 1.

	+	*	(	)	<i>id</i>	\$
+	>	<	<	>	<	>
*	>	>	<	>	<	>
(	<	<	<	=	<	
)	>	>		>		>
<i>id</i>	>	>		>		>
\$	<	<	<		<	

5.1.7. Bottom-Up Parsing

Let $id + ((id + id) * (id)) * id$ be a string obtained by rightmost derivation, then applying Algorithm 1 results in the bottom-up parsing detailed in Table 7.

Table 7. Bottom-up Parsing of Example 1.

Stack	Input	Action
\$	$id + ((id + id) * (id)) * id\$$	Shift
$\$id$	$+((id + id) * (id)) * id\$$	Reduce
\$	$+((id + id) * (id)) * id\$$	Shift
$\$+$	$((id + id) * (id)) * id\$$	Shift
$\$ + ($	$(id + id) * (id)) * id\$$	Shift
$\$ + (($	$id + id) * (id)) * id\$$	Shift
$\$ + ((id$	$+id) * (id)) * id\$$	Reduce
$\$ + (($	$+id) * (id)) * id\$$	Shift
$\$ + ((+$	$id) * (id)) * id\$$	Shift
$\$ + ((+id$	$) * (id)) * id\$$	Reduce
$\$ + ((+$	$) * (id)) * id\$$	Reduce
$\$ + (($	$) * (id)) * id\$$	Shift
$\$ + (($	$*(id)) * id\$$	Reduce
$\$ + ($	$*(id)) * id\$$	Shift
$\$ + (*$	$(id)) * id\$$	Shift
$\$ + (*($	$id)) * id\$$	Shift
$\$ + (*(id$	$)) * id\$$	Reduce
$\$ + (*($	$)) * id\$$	Shift

Table 7. Cont.

Stack	Input	Action
\$ + (*()	) * id\$	Reduce
\$ + (*	) * id\$	Reduce
\$ + (	) * id\$	Shift
\$ + ()	*id\$	Reduce
\$ +	*id\$	Shift
\$ + *	id\$	Shift
\$ + *id	\$	Reduce
\$ + *	\$	Reduce
\$ +	\$	Reduce
\$	\$	Accept

5.2. Example 2

5.2.1. Grammar

Given the following non-OPG:

$$S \rightarrow SD; \mid D;$$

$$D \rightarrow Tid(L)$$

$$T \rightarrow T * \mid int$$

$$L \rightarrow I \mid \varepsilon$$

$$I \rightarrow T \mid T, I$$

The sets are built step by step following the algorithms.

5.2.2. Left

The Left sets are calculated as detailed in Table 8.

Table 8. Left Sets calculation for the CFG in Example 2.

Left(S)			
For $S \rightarrow SD;$			
1. Evaluating the first conditional of Algorithm 2.	$A = S$ $\beta = \varepsilon$ $B = S$ $\gamma = D;$ $\mathcal{L}(S) \supseteq \mathcal{L}(S)$	2. Evaluating the first conditional of Algorithm 2.	$A = S$ $\beta = S$ $B = D$ $\gamma = ;$ $\mathcal{L}(S) \supseteq \mathcal{L}(D)$
3. Evaluating the second conditional of Algorithm 2.	$A = S$ $\beta = SD$ $a = ;$ $\gamma = \varepsilon$ $\mathcal{L}(S) \supseteq \mathcal{L}(D) \cup \{ ; \}$		
For $S \rightarrow D;$			
4. Evaluating the first conditional of Algorithm 2.	$A = S$ $\beta = \varepsilon$ $B = D$ $\gamma = ;$ $\mathcal{L}(S) \supseteq \mathcal{L}(D) \cup \{ ; \} \cup \mathcal{L}(D)$	5. Evaluating the second conditional of Algorithm 2.	$A = S$ $\beta = D$ $a = ;$ $\gamma = \varepsilon$ $\mathcal{L}(S) = \mathcal{L}(D) \cup \{ ; \}$

Table 8. Cont.

Left(D)			
For $D \rightarrow Tid(L)$			
1. Evaluating the first conditional of Algorithm 2.	$A = D$ $\beta = \varepsilon$ $B = T$ $\gamma = id(L)$ $\mathcal{L}(D) \supseteq \mathcal{L}(T)$	2. Evaluating the second conditional of Algorithm 2.	$A = D$ $\beta = T$ $a = id$ $\gamma = (L)$ $\mathcal{L}(D) = \mathcal{L}(T) \cup \{id\}$
Left(T)			
For $T \rightarrow T^*$			
1. Evaluating the first conditional of Algorithm 2.	$A = T$ $\beta = \varepsilon$ $B = T$ $\gamma = *$ $\mathcal{L}(T) \supseteq \mathcal{L}(T)$	2. Evaluating the second conditional of Algorithm 2.	$A = T$ $\beta = T$ $a = *$ $\gamma = \varepsilon$ $\mathcal{L}(T) \supseteq \{*\}$
For $T \rightarrow int$			
3. Evaluating the second conditional of Algorithm 2.	$A = T$ $\beta = \varepsilon$ $a = int$ $\gamma = \varepsilon$ $\mathcal{L}(T) = \{*, int\}$		
Left(L)			
For $L \rightarrow I$			
1. Evaluating the first conditional of Algorithm 2.	$A = L$ $\beta = \varepsilon$ $B = I$ $\gamma = \varepsilon$ $\mathcal{L}(L) = \mathcal{L}(I)$		
Left(I)			
For $I \rightarrow T$			
1. Evaluating the first conditional of Algorithm 2.	$A = I$ $\beta = \varepsilon$ $B = T$ $\gamma = \varepsilon$ $\mathcal{L}(I) \supseteq \mathcal{L}(T)$		
For $I \rightarrow T, I$			
2. Evaluating the first conditional of Algorithm 2.	$A = I$ $\beta = \varepsilon$ $B = T$ $\gamma = ,I$ $\mathcal{L}(I) \supseteq \mathcal{L}(T)$	3. Evaluating the second conditional of Algorithm 2.	$A = I$ $\beta = T$ $a = ,$ $\gamma = I$ $\mathcal{L}(I) = \mathcal{L}(T) \cup \{ , \}$

Summary of Left sets of non-terminals:

- $\text{Left}(S) = \mathcal{L}(D) \cup \{ ; \}$.
- $\text{Left}(D) = \mathcal{L}(T) \cup \{ id \}$.
- $\text{Left}(T) = \{ *, int \}$.
- $\text{Left}(L) = \mathcal{L}(I)$.
- $\text{Left}(I) = \mathcal{L}(T) \cup \{ , \}$.

Substituting the sets:

- $\text{Left}(S) = \{;, id, *, int\}$.
- $\text{Left}(D) = \{id, *, int\}$.
- $\text{Left}(T) = \{*, int\}$.
- $\text{Left}(L) = \{,, *, int\}$.
- $\text{Left}(I) = \{,, *, int\}$.

5.2.3. Right

The Right sets are calculated as detailed in Table 9.

Table 9. Right Sets calculation for the CFG in Example 2.

Right(S)			
For $S \rightarrow SD;$		For $S \rightarrow D;$	
1. Evaluating the second conditional of Algorithm 4.	$A = S$ $\gamma = SD$ $a = ;$ $\beta = \varepsilon$ $\mathcal{R}(S) \supseteq \{;\}$	2. Evaluating the second conditional of Algorithm 4.	$A = S$ $\gamma = D$ $a = ;$ $\beta = \varepsilon$ $\mathcal{R}(S) = \{;\}$
Right(D)			
For $D \rightarrow Tid(L)$			
1. Evaluating the second conditional of Algorithm 4.	$A = D$ $\gamma = Tid(L$ $a =)$ $\beta = \varepsilon$ $\mathcal{R}(D) = \{)\}$		
Right(T)			
For $T \rightarrow T*$		For $T \rightarrow int$	
1. Evaluating the second conditional of Algorithm 4.	$A = T$ $\gamma = T$ $a = *$ $\beta = \varepsilon$ $\mathcal{R}(T) \supseteq \{*\}$	2. Evaluating the second conditional of Algorithm 4.	$A = T$ $\gamma = \varepsilon$ $a = int$ $\beta = \varepsilon$ $\mathcal{R}(T) = \{*, int\}$
Right(L)			
For $L \rightarrow I$			
1. Evaluating the first conditional of Algorithm 4.	$A = L$ $\gamma = \varepsilon$ $B = I$ $\beta = \varepsilon$ $\mathcal{R}(L) = \mathcal{R}(I)$		

Table 9. Cont.

Right(I)			
For $I \rightarrow T$			
1. Evaluating the first conditional of Algorithm 4.	$A = I$ $\gamma = \varepsilon$ $B = T$ $\beta = \varepsilon$ $\mathcal{R}(I) \supseteq \mathcal{R}(T)$		
For $I \rightarrow T, I$			
2. Evaluating the first conditional of Algorithm 4.	$A = I$ $\gamma = T,$ $B = I$ $\beta = \varepsilon$ $\mathcal{R}(I) \supseteq \mathcal{R}(T) \cup \mathcal{R}(I)$	3. Evaluating the second conditional of Algorithm 4.	$A = I$ $\gamma = T$ $a = ,$ $\beta = I$ $\mathcal{R}(I) = \mathcal{R}(T) \cup \{ , \}$

Summary of Right sets of non-terminals:

- $\text{Right}(S) = \{ ; \}$.
- $\text{Right}(D) = \{ \}$.
- $\text{Right}(T) = \{ *, int \}$.
- $\text{Right}(L) = \mathcal{R}(I)$.
- $\text{Right}(I) = \mathcal{R}(T) \cup \{ , \}$.

Substituting the sets:

- $\text{Right}(S) = \{ ; \}$.
- $\text{Right}(D) = \{ \}$.
- $\text{Right}(T) = \{ *, int \}$.
- $\text{Right}(L) = \{ , , *, int \}$.
- $\text{Right}(I) = \{ , , *, int \}$.

5.2.4. Leftmost

The Leftmost sets are calculated as detailed in Table 10.

Table 10. Leftmost Sets calculation for the CFG in Example 2.

Leftmost(S)			
For $S \rightarrow SD;$		For $S \rightarrow D;$	
1. Evaluating the first conditional of Algorithm 3.	$A = S$ $\beta = \varepsilon$ $B = S$ $\gamma = D;$ $\mathcal{L}_m(S) \supseteq \mathcal{L}_m(S)$	2. Evaluating the first conditional of Algorithm 3.	$A = S$ $\beta = \varepsilon$ $B = D$ $\gamma = ;$ $\mathcal{L}_m(S) = \mathcal{L}_m(D)$
Leftmost(D)			
For $D \rightarrow Tid(L)$			
1. Evaluating the first conditional of Algorithm 3.	$A = D$ $\beta = \varepsilon$ $B = T$ $\gamma = id(L)$ $\mathcal{L}_m(D) = \mathcal{L}_m(T)$		

Table 10. Cont.

Leftmost(T)			
For $T \rightarrow T^*$		For $T \rightarrow int$	
1. Evaluating the first conditional of Algorithm 3.	$A = T$ $\beta = \varepsilon$ $B = T$ $\gamma = *$ $\mathcal{L}_m(T) \supseteq \mathcal{L}_m(T)$	2. Evaluating the second conditional of Algorithm 3.	$A = T$ $\beta = \varepsilon$ $a = int$ $\gamma = \varepsilon$ $\mathcal{L}_m(T) = \{int\}$
Leftmost(L)			
For $L \rightarrow I$			
1. Evaluating the first conditional of Algorithm 3.	$A = L$ $\beta = \varepsilon$ $B = I$ $\gamma = \varepsilon$ $\mathcal{L}_m(L) = \mathcal{L}_m(I)$		
Leftmost(I)			
For $I \rightarrow T$		For $I \rightarrow T, I$	
1. Evaluating the first conditional of Algorithm 3.	$A = I$ $\beta = \varepsilon$ $B = T$ $\gamma = \varepsilon$ $\mathcal{L}_m(I) \supseteq \mathcal{L}_m(T)$	2. Evaluating the first conditional of Algorithm 3.	$A = I$ $\beta = \varepsilon$ $B = T$ $\gamma =, I$ $\mathcal{L}_m(I) = \mathcal{L}_m(T)$

Summary of Leftmost sets of non-terminals:

- Leftmost(S) = $\mathcal{L}_m(D)$.
- Leftmost(D) = $\mathcal{L}_m(T)$.
- Leftmost(T) = $\{int\}$.
- Leftmost(L) = $\mathcal{L}_m(I)$.
- Leftmost(I) = $\mathcal{L}_m(T)$.

Substituting the sets:

- Leftmost(S) = $\{int\}$.
- Leftmost(D) = $\{int\}$.
- Leftmost(T) = $\{int\}$.
- Leftmost(L) = $\{int\}$.
- Leftmost(I) = $\{int\}$.

5.2.5. Case Studies

For Algorithm 5, the application of some precedence relations is exhibited.

- For the delimiter symbol \$:
 - $\$ \prec \mathcal{L}(S) \rightarrow \text{Rule (A.1)}.$
 - $\$ \prec ;$
 - $\$ \prec id$
 - $\$ \prec *$
 - $\$ \prec int$
 - $\mathcal{R}(S) \succ \$ \rightarrow \text{Rule (A.2)}.$
 - $; \succ \$$
- For $S \rightarrow SD$;

- (a) $X = S, Y = D \rightarrow \text{Rule (E.1)}.$
 $\mathcal{R}(S) \succ \mathcal{L}_m(D)$
 $\bullet \quad ; \succ int$
 $u = null, l = []$
- (b) $X = D, Y = ; \rightarrow \text{Rule (D.1)}.$
 $\mathcal{R}(D) \succ ;$
 $\bullet \quad) \succ ;$
 $u = null, l = []$
3. For $D \rightarrow Tid(L)$
- (a) $X = id, Y = (\rightarrow \text{Rule (B)}.$
 $\bullet \quad id \doteq ($
 $u = null, l = []$
- (b) $X = (, Y = L \rightarrow \text{Rule (C)}.$
 $(\prec \mathcal{L}(L)$
 $\bullet \quad (\prec ,$
 $\bullet \quad (\prec *$
 $\bullet \quad (\prec int$
 $u = (, l = []$
- (c) $X = L, Y =) \rightarrow \text{Rule (D.1)}.$
 $\mathcal{R}(L) \succ)$
 $\bullet \quad , \succ)$
 $\bullet \quad * \succ)$
 $\bullet \quad int \succ)$
 $u \neq null \rightarrow u \doteq b \rightarrow (\doteq) \rightarrow \text{Rule (D.2)}.$
 $u = null, l = []$
4. In the following cases, the form XY is not fulfilled in the right side part of the production; therefore, no rule can be applied:
- (a) $T \rightarrow int$
(b) $L \rightarrow I$
(c) $L \rightarrow \varepsilon$
(d) $I \rightarrow T$

The remaining precedence relations are derived using the same method as the previously demonstrated cases.

5.2.6. Operator Precedence Table

At the end of the algorithm execution, operator precedence Table 11 is obtained.

Table 11. Precedence Table of Example 2.

	$;$	id	$($	$)$	$*$	int	$,$	$\$$
$;$						$\succ$		$\succ$
id			$\doteq$					
$($				$\doteq$	$\prec$	$\prec$	$\prec$	
$)$	$\succ$							
$*$		$\succ$		$\succ$	$\succ$		$\succ$	
int		$\succ$		$\succ$	$\succ$		$\succ$	
$,$				$\succ$	$\prec$	$\prec$	$\prec$	
$\$$	$\prec$	$\prec$			$\prec$	$\prec$		

5.2.7. Bottom-Up Parsing

Let $int\ id();\ int\ id(int, int);$ be a string obtained by rightmost derivation, then applying Algorithm 1 results in the bottom-up parsing detailed in Table 12.

Table 12. Bottom-up Parsing of Example 2.

Stack	Input	Action
\$	$int\ id();\ int\ id(int, int); \$$	Shift
$\$int$	$id();\ int\ id(int, int); \$$	Reduce
\$	$id();\ int\ id(int, int); \$$	Shift
$\$id$	$();\ int\ id(int, int); \$$	Shift
$\$id($	$);\ int\ id(int, int); \$$	Shift
$\$id()$	$;\ int\ id(int, int); \$$	Reduce
\$	$;\ int\ id(int, int); \$$	Shift
$\$;$	$int\ id(int, int); \$$	Reduce
\$	$int\ id(int, int); \$$	Shift
$\$int$	$id(int, int); \$$	Reduce
\$	$id(int, int); \$$	Shift
$\$id$	$(int, int); \$$	Shift
$\$id($	$int, int); \$$	Shift
$\$id(int$	$, int); \$$	Reduce
$\$id($	$, int); \$$	Shift
$\$id(,$	$int); \$$	Shift
$\$id(, int$	$); \$$	Reduce
$\$id(,$	$); \$$	Reduce
$\$id($	$); \$$	Shift
$\$id()$	$;\ \$$	Reduce
\$	$;\ \$$	Shift
$\$;$	$\$$	Reduce
\$	$\$$	<i>Accept</i>

5.3. Example 3

5.3.1. Grammar

Given the following non-OPG:

$$S \longrightarrow ABC$$

$$A \longrightarrow aA \mid a$$

$$B \longrightarrow bB \mid b \mid \varepsilon$$

$$C \longrightarrow CDc \mid c$$

$$D \longrightarrow d$$

The sets are built step by step following the algorithms.

5.3.2. Left

The Left sets are calculated as detailed in Table 13.

Table 13. Left Sets calculation for the CFG in Example 3.

Left(S)			
For $S \rightarrow ABC$			
1. Evaluating the first conditional of Algorithm 2.	$A = S$ $\beta = \varepsilon$ $B = A$ $\gamma = BC$ $\mathcal{L}(S) \supseteq \mathcal{L}(A)$	2. Evaluating the first conditional of Algorithm 2.	$A = S$ $\beta = A$ $B = B$ $\gamma = C$ $\mathcal{L}(S) \supseteq \mathcal{L}(A) \cup \mathcal{L}(B)$
3. Evaluating the first conditional of Algorithm 2.	$A = S$ $\beta = AB$ $B = C$ $\gamma = \varepsilon$ $\mathcal{L}(S) = \mathcal{L}(A) \cup \mathcal{L}(B) \cup \mathcal{L}(C)$		
Left(A)			
For $A \rightarrow aA$			
1. Evaluating the second conditional of Algorithm 2.	$A = A$ $\beta = \varepsilon$ $a = a$ $\gamma = A$ $\mathcal{L}(A) \supseteq \{a\}$	2. Evaluating the first conditional of Algorithm 2.	$A = A$ $\beta = a$ $B = A$ $\gamma = \varepsilon$ $\mathcal{L}(A) \supseteq \{a\} \cup \mathcal{L}(A)$
For $A \rightarrow a$			
3. Evaluating the second conditional of Algorithm 2.	$A = A$ $\beta = \varepsilon$ $a = a$ $\gamma = \varepsilon$ $\mathcal{L}(A) = \{a\}$		
Left(B)			
For $B \rightarrow bB$			
1. Evaluating the second conditional of Algorithm 2.	$A = B$ $\beta = \varepsilon$ $a = b$ $\gamma = B$ $\mathcal{L}(B) \supseteq \{b\}$	2. Evaluating the first conditional of Algorithm 2.	$A = B$ $\beta = b$ $B = B$ $\gamma = \varepsilon$ $\mathcal{L}(B) \supseteq \{b\} \cup \mathcal{L}(B)$
For $B \rightarrow b$			
3. Evaluating the second conditional of Algorithm 2.	$A = B$ $\beta = \varepsilon$ $a = b$ $\gamma = \varepsilon$ $\mathcal{L}(B) = \{b\}$		

Table 13. Cont.

Left(C)			
For $C \rightarrow CDc$			
1. Evaluating the first conditional of Algorithm 2.	$A = C$ $\beta = \varepsilon$ $B = C$ $\gamma = Dc$ $\mathcal{L}(C) \supseteq \mathcal{L}(C)$	2. Evaluating the first conditional of Algorithm 2.	$A = C$ $\beta = C$ $B = D$ $\gamma = c$ $\mathcal{L}(C) \supseteq \mathcal{L}(D)$
3. Evaluating the second conditional of Algorithm 2.	$A = C$ $\beta = CD$ $a = c$ $\gamma = \varepsilon$ $\mathcal{L}(C) \supseteq \mathcal{L}(D) \cup \{c\}$		
For $C \rightarrow c$			
4. Evaluating the second conditional of Algorithm 2.	$A = C$ $\beta = \varepsilon$ $a = c$ $\gamma = \varepsilon$ $\mathcal{L}(C) = \mathcal{L}(D) \cup \{c\}$		
Left(D)			
For $D \rightarrow d$			
1. Evaluating the second conditional of Algorithm 2.	$A = D$ $\beta = \varepsilon$ $a = d$ $\gamma = \varepsilon$ $\mathcal{L}(D) = \{d\}$		

Summary of Left sets of non-terminals:

- $\text{Left}(S) = \mathcal{L}(A) \cup \mathcal{L}(B) \cup \mathcal{L}(C)$.
- $\text{Left}(A) = \{a\}$.
- $\text{Left}(B) = \{b\}$.
- $\text{Left}(C) = \mathcal{L}(D) \cup \{c\}$.
- $\text{Left}(D) = \{d\}$.

Substituting the sets:

- $\text{Left}(S) = \{a, b, d, c\}$.
- $\text{Left}(A) = \{a\}$.
- $\text{Left}(B) = \{b\}$.
- $\text{Left}(C) = \{d, c\}$.
- $\text{Left}(D) = \{d\}$.

5.3.3. Right

The Right sets are calculated as detailed in Table 14

Table 14. Right Sets calculation for the CFG in Example 3.

Right(S)			
For $S \rightarrow ABC$			
1. Evaluating the first conditional of Algorithm 4.	$A = S$ $\gamma = AB$ $B = C$ $\beta = \varepsilon$ $\mathcal{R}(S) = \mathcal{R}(C)$		
Right(A)			
For $A \rightarrow aA$			
1. Evaluating the first conditional of Algorithm 4.	$A = A$ $\gamma = a$ $B = A$ $\beta = \varepsilon$ $\mathcal{R}(A) \supseteq \mathcal{R}(A)$	2. Evaluating the second conditional of Algorithm 4.	$A = A$ $\gamma = \varepsilon$ $a = a$ $\beta = A$ $\mathcal{R}(A) \supseteq \{a\}$
For $A \rightarrow a$			
3. Evaluating the second conditional of Algorithm 4.	$A = A$ $\gamma = \varepsilon$ $a = a$ $\beta = \varepsilon$ $\mathcal{R}(A) = \{a\}$		
Right(B)			
For $B \rightarrow bB$			
1. Evaluating the first conditional of Algorithm 4.	$A = B$ $\gamma = b$ $B = B$ $\beta = \varepsilon$ $\mathcal{R}(B) \supseteq \mathcal{R}(B)$	2. Evaluating the second conditional of Algorithm 4.	$A = B$ $\gamma = \varepsilon$ $a = b$ $\beta = B$ $\mathcal{R}(B) \supseteq \{b\}$
For $B \rightarrow b$			
3. Evaluating the second conditional of Algorithm 4.	$A = B$ $\gamma = \varepsilon$ $a = b$ $\beta = \varepsilon$ $\mathcal{R}(b) = \{b\}$		
Right(C)			
For $C \rightarrow CDc$		For $C \rightarrow c$	
1. Evaluating the second conditional of Algorithm 4.	$A = C$ $\gamma = CD$ $a = c$ $\beta = \varepsilon$ $\mathcal{R}(C) \supseteq \{c\}$	2. Evaluating the second conditional of Algorithm 4.	$A = C$ $\gamma = \varepsilon$ $a = c$ $\beta = \varepsilon$ $\mathcal{R}(C) = \{c\}$

Table 14. Cont.

Right(D)	
For $D \rightarrow d$	
1. Evaluating the second conditional of Algorithm 4.	$A = D$ $\gamma = \varepsilon$ $a = d$ $\beta = \varepsilon$ $\mathcal{R}(D) = \{d\}$

Summary of Right sets of non-terminals:

- $\text{Right}(S) = \mathcal{R}(C)$.
- $\text{Right}(A) = \{a\}$.
- $\text{Right}(B) = \{b\}$.
- $\text{Right}(C) = \{c\}$.
- $\text{Right}(D) = \{d\}$.

Substituting the sets:

- $\text{Right}(S) = \{c\}$.
- $\text{Right}(A) = \{a\}$.
- $\text{Right}(B) = \{b\}$.
- $\text{Right}(C) = \{c\}$.
- $\text{Right}(D) = \{d\}$.

5.3.4. Leftmost

The Leftmost sets are calculated as detailed in Table 15.

Table 15. Leftmost Sets calculation for the CFG in Example 3.

Leftmost(S)			
For $S \rightarrow ABC$			
1. Evaluating the first conditional of Algorithm 3.	$A = S$ $\beta = \varepsilon$ $B = A$ $\gamma = BC$ $\mathcal{L}_m(S) = \mathcal{L}_m(A)$		
Leftmost(A)			
For $A \rightarrow aA$		For $A \rightarrow a$	
1. Evaluating the second conditional of Algorithm 3.	$A = A$ $\beta = \varepsilon$ $a = a$ $\gamma = A$ $\mathcal{L}_m(A) \supseteq \{a\}$	2. Evaluating the second conditional of Algorithm 3.	$A = A$ $\beta = \varepsilon$ $a = a$ $\gamma = \varepsilon$ $\mathcal{L}_m(A) = \{a\}$
Leftmost(B)			
For $B \rightarrow bB$		For $B \rightarrow b$	
1. Evaluating the second conditional of Algorithm 3.	$A = B$ $\beta = \varepsilon$ $a = b$ $\gamma = B$ $\mathcal{L}_m(B) \supseteq \{b\}$	2. Evaluating the second conditional of Algorithm 3.	$A = B$ $\beta = \varepsilon$ $a = b$ $\gamma = \varepsilon$ $\mathcal{L}_m(B) = \{b\}$

Table 15. Cont.

Leftmost(C)			
For $C \rightarrow CDc$		For $C \rightarrow c$	
1. Evaluating the first conditional of Algorithm 3.	$A = C$ $\beta = \varepsilon$ $B = C$ $\gamma = Dc$ $\mathcal{L}_m(C) \supseteq \mathcal{L}_m(C)$	2. Evaluating the second conditional of Algorithm 3.	$A = C$ $\beta = \varepsilon$ $a = c$ $\gamma = \varepsilon$ $\mathcal{L}_m(C) = \{c\}$
Leftmost(D)			
For $D \rightarrow d$			
1. Evaluating the second conditional of Algorithm 3.	$A = D$ $\beta = \varepsilon$ $a = d$ $\gamma = \varepsilon$ $\mathcal{L}_m(D) = \{d\}$		

Summary of Leftmost sets of non-terminals:

- $\text{Leftmost}(S) = \mathcal{L}_m(A)$.
- $\text{Leftmost}(A) = \{a\}$.
- $\text{Leftmost}(B) = \{b\}$.
- $\text{Leftmost}(C) = \{c\}$.
- $\text{Leftmost}(D) = \{d\}$.

Substituting the sets:

- $\text{Leftmost}(S) = \{a\}$.
- $\text{Leftmost}(A) = \{a\}$.
- $\text{Leftmost}(B) = \{b\}$.
- $\text{Leftmost}(C) = \{c\}$.
- $\text{Leftmost}(D) = \{d\}$.

5.3.5. Case Studies

For Algorithm 5, the application of some precedence relations is exhibited.

- For the delimiter symbol \$:
 - $\$ \prec \mathcal{L}(S) \rightarrow \text{Rule (A.1)}.$
 - $\$ \prec a$
 - $\$ \prec b$
 - $\$ \prec d$
 - $\$ \prec c$
 - $\mathcal{R}(S) \succ \$ \rightarrow \text{Rule (A.2)}.$
 - $c \succ \$$
- For $S \rightarrow ABC$
 - $X = A, Y = B \rightarrow \text{Rule (E.1)}.$
 $\mathcal{R}(A) \succ \mathcal{L}_m(B)$
 - $a \succ b$ $u = \text{null}, l = [A]$
 - $X = B, Y = C \rightarrow \text{Rule (E.1)}.$
 $\mathcal{R}(B) \succ \mathcal{L}_m(C)$
 - $b \succ c$

- $$A \in l \neq [] \rightarrow \text{Rule (E.3).}$$
- $$\mathcal{R}(A) \succ \mathcal{L}_m(C)$$
- $a \succ c$
- $$u = \text{null}, l = []$$
3. For $A \rightarrow aA$
 - (a) $X = a, Y = A \rightarrow \text{Rule (C).}$
 $a \prec \mathcal{L}(A)$
 - $a \prec a$ $u = a, l = []$
 4. For $B \rightarrow bB$
 - (a) $X = b, Y = B \rightarrow \text{Rule (C).}$
 $b \prec \mathcal{L}(B)$
 - $b \prec b$ $u = b, l = []$
 5. For $C \rightarrow CDc$
 - (a) $X = C, Y = D \rightarrow \text{Rule (E.1).}$
 $\mathcal{R}(C) \succ \mathcal{L}_m(D)$
 - $c \succ d$ $u = \text{null}, l = []$
 - (b) $X = D, Y = c \rightarrow \text{Rule (D.1).}$
 $\mathcal{R}(D) \succ c$
 - $d \succ c$ $u = \text{null}, l = []$
 6. In the following cases, the form XY is not fulfilled in the right side part of the production; therefore, no rule can be applied:
 - (a) $A \rightarrow a$
 - (b) $B \rightarrow b$
 - (c) $B \rightarrow \varepsilon$
 - (d) $C \rightarrow c$
 - (e) $D \rightarrow d$

The remaining precedence relations are derived using the same method as the previously demonstrated cases.

5.3.6. Operator Precedence Table

At the end of the algorithm execution, operator precedence Table 16 is obtained.

Table 16. Precedence Table of Example 3.

	a	b	c	d	$\$$
a	$\prec$	$\succ$	$\succ$		
b		$\prec$	$\succ$		
c				$\succ$	$\succ$
d			$\succ$		
$\$$	$\prec$	$\prec$	$\prec$	$\prec$	

5.3.7. Bottom-Up Parsing

Let $aabbcdc$ be a string obtained by rightmost derivation, then applying Algorithm 1 results in the bottom-up parsing detailed in Table 17.

Table 17. Bottom-up Parsing of Example 3.

Stack	Input	Action
\$	aabbcdc\$	Shift
\$a	abbcdc\$	Shift
\$aa	bbcdc\$	Reduce
\$a	bbcdc\$	Reduce
\$	bbcdc\$	Shift
\$b	bcdc\$	Shift

Table 17. Cont.

Stack	Input	Action
\$bb	cdc\$	Reduce
\$b	cdc\$	Reduce
\$	cdc\$	Shift
\$c	dc\$	Reduce
\$	dc\$	Shift
\$d	c\$	Reduce
\$	c\$	Shift
\$c	\$	Reduce
\$	\$	Accept

5.4. Discussion

In the examples, three CFGs are considered: one OPG and two non-OPGs. In the first CFG, it is observed that it complies with the basic rule, while the following ones do not since they present productions with consecutive non-terminal symbols on the right side. Regardless of the nature of the CFG, the proposed algorithms play a crucial role in obtaining the operator precedence table. Using Algorithm 1 applied to a specific string for each CFG, a bottom-up parsing is performed, verifying that the strings are generated by their respective grammars. This process underscores the functionality of the proposal to perform bottom-up parsing based on operator precedence, even when the CFGs do not fulfill the basic requirement of the OPGs.

6. Exceptions

If the CFG has any of the following forms, the rules outlined in the algorithms of this paper would not apply:

1. $A \rightarrow aBb$
 $B \rightarrow \beta a$

The sets for B in the second production are:

$$\mathcal{L}(B) = \{a\}.$$

$$\mathcal{R}(B) = \{a\}.$$

The precedence relations based on the sections of Algorithm 5 exhibit:

- For the rule (C): $a \leq a$ and $u = a$.
- For the rule (D.1): $a > b$.
- For the rule (D.2): $a \doteq b$.

Applying rules (D.1) and (D.2) generates the conflict.

2. $A \longrightarrow aBb$
 $B \longrightarrow a\beta$
 Being $\beta \xRightarrow{*} \varepsilon \vee \beta \xRightarrow{*} \gamma a$.
 According to the established conditions, the sets for B in the second production are
 $\mathcal{L}(B) = \{a\}$.
 $\mathcal{R}(B) = \{a\}$.
 The precedence relations based on the sections of Algorithm 5 exhibit:
- For the rule (C): $a \leq a$ and $u = a$.
 - For the rule (D.1): $a \geq b$.
 - For the rule (D.2): $a \doteq b$.
- The application of rules (D.1) and (D.2) generates the conflict.
 If the $\mathcal{R}(B)$ does not contain a , then the conflict does not arise.
3. $A \longrightarrow bBa$
 $B \longrightarrow a\beta$
 The sets for B in the second production are:
 $\mathcal{L}(B) = \{a\}$.
 $\mathcal{R}(B) = \{a\}$.
 The precedence relations based on the sections of Algorithm 5 exhibit:
- For the rule (C): $b \leq a$ and $u = b$.
 - For the rule (D.1): $a \geq a$.
 - For the rule (D.2): $b \doteq a$.
- The application of rules (C) and (D.2) generates the conflict.
4. $A \longrightarrow bBa$
 $B \longrightarrow \beta a$
 Being $\beta \xRightarrow{*} \varepsilon \vee \beta \xRightarrow{*} a\gamma$.
 According to the established conditions, the sets for B in the second production are:
 $\mathcal{L}(B) = \{a\}$.
 $\mathcal{R}(B) = \{a\}$.
 The precedence relations based on the sections of Algorithm 5 exhibit:
- For the rule (C): $b \leq a$ and $u = b$.
 - For the rule (D.1): $a \geq a$.
 - For the rule (D.2): $b \doteq a$.
- The application of rules (C) and (D.2) generates the conflict.
 If the $\mathcal{L}(B)$ does not contain a , then the conflict does not arise.

When CFG productions take the following form, there will be no conflicts:

$$A \longrightarrow aBb$$

$$B \longrightarrow \gamma$$

where γ has neither a at the end of the right side nor b at the end of the left side.

7. Conclusions

Previous work has addressed the application of OPGs to solve problems related to the parsing of various forms of languages; however, not much emphasis has been placed on the definition of the Left and Right sets nor on the generation of an algorithm to find the operator precedence table.

In this work, we have redefined the sets Right and Left, previously exposed by other authors, and introduced a novel set called Leftmost. These sets form the basis for an algorithm that constructs the operator precedence table. Each set is constructed from an algorithm that facilitates its obtaining, adding a new dimension to the existing research.

The proposed algorithm for constructing the precedence table, while breaking (with some exceptions) the basic definition of an OPG, opens up new possibilities. It allows for the construction of an operator precedence table from a non-OPG CFG, paving the way for a bottom-up parsing algorithm to recognize a string generated by the CFG.

Three examples of CFGs are shown: one OPG and two non-OPGs. The proposed algorithms are systematically applied to these examples, step by step, until the operator precedence tables are obtained, regardless of the CFG's nature. Additionally, the bottom-up parsing algorithm is applied to specific strings for each grammar, providing verification that they generated them.

It is important to note that while the algorithm and the definition of the new sets of terminal symbols are significant advancements, they do have limitations. It is not always possible to solve all cases where two or more adjacent terminal symbols are on the right-hand side. There are exceptions, and we must be aware of them.

Author Contributions: Conceptualization, L.L. and J.M.; methodology, J.M.; software, L.L.; validation, L.L. and J.M.; formal analysis, L.L., E.A., and J.M.; investigation, L.L. and J.M.; resources, J.M.; data curation, E.A.; writing—original draft preparation, E.A. and J.M.; writing—review and editing, E.A. and J.M.; visualization, E.A.; supervision, J.M.; project administration, J.M.; funding acquisition, J.M.; All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding and the APC was funded by Universidad del Norte.

Data Availability Statement: Data sharing is not applicable to this article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Floyd, R.W. Syntactic analysis and operator precedence. *J. ACM (JACM)* **1963**, *10*, 316–333. [\[CrossRef\]](#)
2. Alfred, V.; Monica, S.; Ravi, S.; Jeffrey, D.U. *Compilers Principles, Techniques*; Pearson Education: London, UK, 2007.
3. Grune, D.; Jacobs, C.J.H. *Parsing Techniques (Monographs in Computer Science)*; Springer: Berlin, Heidelberg, 2006.
4. Crespi-Reghizzi, S. An effective model for grammar inference. In *Information Processing 71*; Gilchrist, B., Ed.; Elsevier: North-Holland, New York, 1972; pp. 524–529.
5. Crespi-Reghizzi, S. Reduction of Enumeration in Grammar Acquisition. In Proceedings of the 2nd International Joint Conference on Artificial Intelligence, San Francisco, CA, USA, 1–3 September 1971; pp. 546–552.
6. Crespi-Reghizzi, S.; Melkanoff, M.A.; Lichten, L. The Use of Grammatical Inference for Designing Programming Languages. *Commun. ACM* **1973**, *16*, 83–90. [\[CrossRef\]](#)
7. von Braunmühl, B.; Verbeek, R. Input Driven Languages are Recognized in log n Space. In *Topics in the Theory of Computation*; North-Holland Mathematics Studies; Karpinski, M., van Leeuwen, J., Eds.; Elsevier: North-Holland, The Netherlands, 1985; Volume 102, pp. 1–19.
8. Mehlhorn, K. Pebbling mountain ranges and its application to DCFL-recognition. In *Automata, Languages and Programming*; de Bakker, J., van Leeuwen, J., Eds.; Springer: Berlin/Heidelberg, Germany, 1980; pp. 422–435.
9. Mandrioli, D.; Pradella, M. Generalizing input-driven languages: Theoretical and practical benefits. *Comput. Sci. Rev.* **2018**, *27*, 61–87. [\[CrossRef\]](#)
10. Alur, R.; Madhusudan, P. Adding Nesting Structure to Words. *J. ACM* **2009**, *56*, 1–43. [\[CrossRef\]](#)
11. Crespi Reghizzi, S.; Mandrioli, D. Operator precedence and the visibly pushdown property. *J. Comput. Syst. Sci.* **2012**, *78*, 1837–1867. [\[CrossRef\]](#)
12. Crespi Reghizzi, S.; Pradella, M. Beyond operator-precedence grammars and languages. *J. Comput. Syst. Sci.* **2020**, *113*, 18–41. [\[CrossRef\]](#)
13. Barengi, A.; Crespi Reghizzi, S.; Mandrioli, D.; Panella, F.; Pradella, M. Parallel parsing made practical. *Sci. Comput. Program.* **2015**, *112*, 195–226. [\[CrossRef\]](#)
14. Crespi-Reghizzi, S.; Mandrioli, D.; Martin, D.F. Algebraic properties of operator precedence languages. *Inf. Control* **1978**, *37*, 115–133. [\[CrossRef\]](#)
15. Chomsky, N. Three models for the description of language. *IRE Trans. Inf. Theory* **1956**, *2*, 113–124. [\[CrossRef\]](#)
16. Aho, A.V.; Ullman, J.D. *The Theory of Parsing, Translation, and Compiling*; Prentice-Hall Englewood: Cliffs, NJ, USA, 1973; Volume 1.
17. Loudon Kenneth, C. Compiler Construction: Principles and Practice. In *Course Technology*; PWS Publishing Co.: Montgomery, IL, USA, 1997.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.

Article

EEG Channel Selection for Stroke Patient Rehabilitation Using BAT Optimizer

Mohammed Azmi Al-Betar ^{1,*} , Zaid Abdi Alkareem Alyasseri ^{2,3,*} , Noor Kamal Al-Qazzaz ⁴,
Sharif Naser Makhadmeh ^{1,5}, Nabeel Salih Ali ²  and Christoph Guger ⁶ 

- ¹ Artificial Intelligence Research Center (AIRC), College of Engineering and Information Technology, Ajman University, Ajman P.O. Box 346, United Arab Emirates; sharif.makhadmeh@uop.edu.jo
 - ² Information Technology Research and Development Center (ITRDC), University of Kufa, Najaf 54001, Iraq; nabeel@uokufa.edu.iq
 - ³ College of Engineering, University of Warith Al-Anbiyaa, Karbala 56001, Iraq
 - ⁴ Biomedical Engineering Department, AL-Khwarizmi College of Engineering, University of Baghdad, Baghdad 47146, Iraq; noorbme@kecbu.uobaghdad.edu.iq
 - ⁵ Data Science and Artificial Intelligence Department, Faculty of Information Technology, University of Petra, Amman 1196, Jordan
 - ⁶ G.Tec Medical Engineering GmbH, 4521 Schiedlberg, Austria; guger@gtec.at
- * Correspondence: m.albetar@ajman.ac.ae (M.A.A.-B.); zaid.alyasseri@uokufa.edu.iq (Z.A.A.A.)

Abstract: Stroke is a major cause of mortality worldwide, disrupts cerebral blood flow, leading to severe brain damage. Hemiplegia, a common consequence, results in motor task loss on one side of the body. Many stroke survivors face long-term motor impairments and require great rehabilitation. Electroencephalograms (EEGs) provide a non-invasive method to monitor brain activity and have been used in brain–computer interfaces (BCIs) to help in rehabilitation. Motor imagery (MI) tasks, detected through EEG, are pivotal for developing BCIs that assist patients in regaining motor purpose. However, interpreting EEG signals for MI tasks remains challenging due to their complexity and low signal-to-noise ratio. The main aim of this study is to focus on optimizing channel selection in EEG-based BCIs specifically for stroke rehabilitation. Determining the most informative EEG channels is crucial for capturing the neural signals related to motor impairments in stroke patients. In this paper, a binary bat algorithm (BA)-based optimization method is proposed to select the most relevant channels tailored to the unique neurophysiological changes in stroke patients. This approach is able to enhance the BCI performance by improving classification accuracy and reducing data dimensionality. We use time–entropy–frequency (TEF) attributes, processed through automated independent component analysis with wavelet transform (AICA-WT) denoising, to enhance signal clarity. The selected channels and features are proved through a k-nearest neighbor (KNN) classifier using public BCI datasets, demonstrating improved classification of MI tasks and the potential for better rehabilitation outcomes.

Keywords: EEG; feature extraction; channel selection; stroke patients; BAT algorithm



Citation: Al-Betar, M.A.; Alyasseri, Z.A.A.; Al-Qazzaz, N.K.; Makhadmeh, S.N.; Ali, N.S.; Guger, C. EEG Channel Selection for Stroke Patient Rehabilitation Using BAT Optimizer. *Algorithms* **2024**, *17*, 346. <https://doi.org/10.3390/a17080346>

Academic Editor: Gabriella Trucco

Received: 21 May 2024

Revised: 14 July 2024

Accepted: 15 July 2024

Published: 8 August 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The World Health Organization (WHO) lists stroke as the second greatest cause of mortality globally because of the damage it causes to the brain through disruptions in cerebral circulation [1,2]. One of the most debilitating forms of nervous system damage is hemiplegia, which describes partial or complete paralysis of one side of the body including the arm, leg, foot, and hand. Ischemic stroke, the most common type of stroke, causes interruption of cerebral perfusion, which results in rapid loss of brain function [3,4]. Sixty percent or more of stroke survivors require rehabilitation due to permanent motor function impairment [5]. However, stroke survivors endure a wide range of disabilities, including visual and cognitive deficits, that have a cumulatively devastating effect on their ability to

carry out even the most basic of tasks. Accordingly, research into efficient treatment and rehabilitation of stroke victims has been a focus for many years [6].

An electroencephalogram (EEG) is a non-invasive method that estimates the electrical activity of the brain with good time resolution in order to define the results of a variety of mental tasks. By translating brain EEG signals into control instructions, brain-computer interfaces (BCIs) help people with physical impairments interact with the sensory world outside [7]. Motor imagery (MI) is the mental imagining of body movement without real muscle movement, and its accompanying rhythmic activities of the brain can be recorded and used as the input signals of BCI systems. Event-related desynchronization/synchronization (ERD/ERS) refers to the detection of the rhythmic power changes in the sensorimotor area within the Rolandic mu (9–13 Hz) and beta (13–30 Hz) frequency bands, which can be used to distinguish between various types of MI tasks [8]. Due to the fact that MI does not require any extraneous stimuli, BCIs based on MI have a wide variety of potential uses [9].

The volatility and complexity of EEG signals, however, cause difficulties for conventional MI-BCI decoding [7]. Crucial and difficult issues remain in enhancing EEG decoding capacity, extracting discriminative information from low signal-to-noise EEGs, and realizing reliable classification of various MI tasks [8].

Channel selection in EEG-based investigations is important to determine the subset of channels that is most informative in terms of capturing the proper neural signals for the chosen application [10]. Nevertheless, such considerations and their corresponding challenges may differ in other application contexts, and that can be seen when comparing stroke rehabilitation with emotion processing and dementia detection using an EEG-based dataset [11]. For example, stroke rehabilitation is characterized by the fact that with neurological defects resulting from the stroke event, there are certain modifications in the activity of the brain and the functioning of the neural networks. EEG data can also be characterized by specific features that distinguish them from healthy subjects or other patients, such as changes in the topological organization of networks, aberrant activity, and appearance of novel regulatory strategies [12].

Therefore, the approach to choosing the most significant EEG channels for stroke rehabilitation may need to be more selective and specific. The idea is to find out which of the channels provides the best recording of the neural characteristics of the particular motor, cognitive, or behavioral impairment which a stroke patient may present, and it will not be the same as the channels that would be ideal for other applications [13]. However, there are distinct approaches such as emotion processing and dementia detection that may employ different neurophysiological mechanisms and/or spectrotemporal patterns. For example, in emotion processing, the frequency band related to emotion processing can be spread across different cortical areas, and the aim of the channel selection could be to capture this distributed frequency activity. Similarly, in the case of dementia detection, the alteration of brain structure and function which define cognitive impairment might represent a different spatial distribution of load that calls for different rules for choosing the channels [14]. With a direct focus on the channel selection problem in the context of stroke rehabilitation, the proposed study intends to contribute an effective solution that can improve the performance of a BCI and thus improve the rehabilitation process.

Channel selection is the main focus of this study. Many electrodes are utilized in medical and diagnostic treatments. For practical BCI applications, many electrodes are needed for classification accuracy; however, putting many electrodes on the scalp is time-consuming and subject to EEG signal overfitting. To fix these issues, researchers need to identify the electrodes that are unable to be classified. Electrode selection can evaluate the neurological knowledge of individuals due to varying reactions and subject-dependent ideal electrode placement [15]. The user-dependent classification job requires automatic electrode relevance determination. This challenge can be overcome by considering a wide range of electrodes and using many methods to identify the appropriate channels for each patient. A particle BA-based optimization technique was used to pick the most accurate

EEG channels for stroke patient rehabilitation. Each EEG data channel provides unique classification features. Channel selection reduces data storage and processing, speeds up classifier training by using simpler data, and reduces the “curse of dimensionality”.

Many studies have examined channel selection in healthy people [16,17], whereas the present study focuses on stroke rehabilitation, where stroke patients’ underlying neurophysiological abnormalities and limitations may demand a customized channel selection strategy.

Choosing EEG channels for emotion processing, cognitive assessment, and clinical diagnosis often involves knowledge of neural mechanisms and brain regions linked with the desired activities. Common brain activity patterns and optimal channel selection are well established for healthy people. However, strokes can alter brain structure, connectivity, and function. These alterations may cause abnormal brain patterns and compensatory processes [18,19]. Thus, the best EEG channels for stroke rehabilitation may differ from those recommended for healthy people.

To capture stroke patients’ neural correlates of motor, cognitive, and behavioral abnormalities, channel selection must be more specialized and personalized due to stroke-induced neuroplasticity. Targeting stroke patients with a customized solution that can properly recognize neural fingerprints and brain activity patterns is intended to connect stroke-related deficiencies. This stage is crucial to developing BCI-based stroke therapies that improve rehabilitation outcomes [20].

The literature on channel selection in healthy people can be useful, but this study recognizes the need to address stroke survivors’ unique issues and considerations. Scientists examined channel selection differences between stroke patients and healthy people to better understand neurophysiological changes and optimal BCI setups for stroke rehabilitation.

Therefore, in this study, a binary bat algorithm (BA)-based optimization method is used to identify the most important EEG channels for stroke rehabilitation. Thus, this method is suited to stroke survivors’ needs by optimizing BCI technology and rehabilitation effectiveness.

Raw EEG data are segmented and filtered using standard filters and the automated independent component analysis with wavelet transform (AICA-WT) denoising approach in an effort to address the issues mentioned above. Then, time–entropy–frequency (TEF) attributes are created by combining the effective features from the time, entropy, and frequency domains. As a result, the BA-based optimization method is used to optimize the TEF characteristics in order to choose the efficient channels that improve the stationarity and resilience of the system. After we train a KNN classifier technique, we conduct a series of experiments to evaluate our proposed framework using 25 MI-based BCI sessions with follow-up assessment visits to examine the functional changes before and after EEG neurorehabilitation from public datasets from the BCI Competition. The hybrid time–entropy–frequency (TEF) attributes used in the AICA-WT-TEF-Chs framework for MI-BCI classification are intended to efficiently leverage the underlying information of the time, entropy, and frequency domains on classification performance.

The approach suggested in this paper makes the following contributions. First, it looks at how time–entropy–frequency (TEF) variables affect classification performance. Second, it uses the effective EEG channels generated by a BA-based optimization method to detect changes both before and after rehabilitation. Third, it tests the effectiveness of the AICA-WT-TEF-Chs framework utilizing a variety of classification models and cutting-edge techniques. By eliminating extraneous dimensions, effective channel selection decreases the amount of data that must be stored and processed, speeds up classifier training by using simpler data, and prevents classifier overfitting by lessening the impact of the “curse of dimensionality”.

2. Related Works

EEG-based BCIs in particular have good temporal resolution [21], but the recorded wave activity may be corrupted in a variety of ways depending on the artifacts that occurred, so preprocessing the raw EEG signals is critical in the classification of MI-based signals.

Most of the time, these artifacts can imitate or overlay the abnormal behavior of the brain. Studies have investigated artifacts' effects on EEG signals, including eye blinks, ocular movements, cardiac artifacts, and muscular activity, and noise from power lines can also intersect an EEG's frequencies [22,23]. As a result, assessing the efficiency of EEG signals in the presence of background noise can be difficult. Spatial filters can produce a more localized signal for each electrode in this situation because EEG-based BCIs in particular have poor spatial resolution.

To choose the best characteristics for BCI-MI-based EEG analysis, feature extraction is carried out for each recorded channel in multiple domains. The most popular technique is to use common spatial patterns (CSPs) to obtain discriminant features from high-dimensional EEG signals, which may be used to identify the spatial characteristics related to ERD produced by various MI tasks [24]. However, the efficiency of CSPs is easily impacted by noise and is highly sensitive to the choice of frequency bands [16]. Several feature extraction techniques were developed to analyze the EEG data in the time, frequency, and time–frequency domains to address the issue of efficacy [25]. Additionally, wavelet coherence (WC) bispectrum characteristics were suggested in [26] for distinguishing between right and left MI.

The sizes of the features derived from EEG signals are usually pretty big for each channel, and they get bigger when one moves to the next channel. The classification of several features necessitates additional computation and time. To solve this issue, the most effective feature set must be obtained by selecting a subset of EEG channels that are more closely related to mental work than others. The four evaluation methodologies of filtering, wrapping, embedding, and hybrid can be used to categorize feature selection algorithms [27]. Based on specific statistical criteria, the filtering method used for motor intention-based EEG activities is taken into consideration. For instance, before classifying the motor intent activity, He et al. [28] developed the statistical method of the Bhattacharyya bound for channel selection. The best candidate subset was chosen using this method, which also employed a CSP and sequential search strategy. Tam et al. [29] suggested a different statistical method based on CSP rank for sorting filter coefficients with absolute values and then choosing the features with the biggest succeeding coefficient. Although the filtering technique has low accuracy, it operates quickly and is unaffected by the subject or classifier choice.

On the other hand, the classifier and the subject are quite important in the wrapper selection process [30]. In this way, the subset candidates are assessed using classification accuracy and can thus make more accurate predictions than filtering techniques. As a result, the wrapper strategy is more computationally expensive than the filtering technique and is subject to the overfitting problem [31], which can be avoided by employing cross-validation measures for prediction. For choosing the best channel, the majority of wrapper strategies use the sequential forward search, backward elimination strategy [32], and heuristic search method [33,34]. Without mentioning a stopping condition, the filtering and wrapping procedures have also been coupled for selection purposes. This hybrid strategy was created to deal with huge datasets. Gaur et al., for instance [35], offered the best possible channel selection technique based on CSPs. Before ranking all the channels according to their scores and using the classification accuracy to evaluate the chosen ideal channel combination, they first applied the L1 norm of a CSP to group the channel contribution scores. Finally, using the defined criteria for a particular classifier's learning process, the embedded approach was used to choose significant channels.

To choose channels with the best classification outcomes, Shi et al. [36] used feature selection techniques, recursive feature removal, and zero-norm optimization trained with support vector machines. The use of these techniques allowed the EEG channels to be significantly reduced [37].

The crucial step in the classification of EEG data is to use pertinent features that significantly affect the system's ability to classify data effectively [38–40]. Therefore, to achieve the best classification performance, it is necessary to use valid features from EEG signals [41].

Furthermore, global optimization techniques and machine learning classification can be combined to address such problems in the BCI domain, including feature extraction and selection [42]. Several algorithms gained a lot of interest for their applications in feature selection, for instance, differential evolution (DE), simulated annealing (SA), particle swarm optimization (PSO), artificial bee colony (ABC), and ant colony optimization (ACO) [43].

Research has been conducted on channel selection for different BCI paradigms including recursive channel elimination for motor imagery tasks [44]. In a study that was conducted by Rakotomamonjy et al. [45], the researchers used a channel selection technique that was built into the training of an SVM classifier. The space of features was searched using GA in the references [46,47], and the fitness function was the weighted linear combination of the number of features and the accuracy of the SVM. Udhaya et al. in [48] suggested a PSO-based rough set feature selection approach to determine the best subset of features, and the accuracy of a neighborhood classifier was utilized as the evaluation criterion for the feature subset. DE was utilized by Baig et al. [49], and PSO and GA were utilized by Atyabi et al. [50] in order to find the best feature subset, respectively. In [51], the process of selecting characteristics for EEG-based emotion identification used ant colony optimization, simulated annealing, genetic algorithms, particle swarm optimization, and differential evolution. The performance of these EA-based approaches for feature selection is quite encouraging.

However, the size of the search area is predetermined in these algorithms; therefore, throughout each iteration, it is important to make a decision regarding whether or not each feature should be chosen. Under these circumstances, the computational effort will be wasted on some features that are superfluous, unnecessary, or of a trivial nature. In addition, evolutionary algorithms (EAs) readily converge to a local optimum as a result of the wide search space as well as the interference of redundant, irrelevant, and trivial features. In this study, we design EA-based feature selection methods for motor imagery BCI through the reduction of dimensionality.

The performance of a classifier will dramatically decline as the number of features increases due to the dimensionality curse, which occurs when the number of features surpasses a specific threshold. Additionally, the training procedure takes longer the higher the feature dimensionality becomes [46].

In order to locate a large number of existing channels, feature selection is required with a thorough search for extracting related channels [10,52]. Researchers have focused on feature selection issues and suggested complete search-, greedy search-, and random search-based approaches [11]. High computational costs and convergence to local optimum remain issues with these selection techniques [53].

3. BAT Algorithm

In this section, the bat optimization algorithm is introduced and illustrated in terms of its inspiration, microbats, and their lifestyle in nature, along with the method's general optimization procedure and mathematical formulation.

3.1. Inspiration of BAT

The bat optimization algorithm was first proposed by Xin-She Yang in 2010 [54]. Xin-She Yang proposed the bat algorithm based on microbats' characteristics of echolocation, used for hunting prey, and movement in darkness. A microbat produces extremely loud sound pulses and hears their echo from the objects and prey around it. Based on the echo sound, bats can define the type of prey and the strategy for hunting it. The bats usually produce 10 to 20 pulses every second when they are searching for prey. Once they are close to the prey, the number of pulses increases to 200 pulses every second and the sound pulse becomes quieter.

In nature, microbats have a collective hunting behavior, where they fly together with determined velocity to hunt prey simultaneously by using the echolocation characteristic, as shown in Figure 1. The figure presents the movement of microbats based on the echoloca-

tion mechanism, where the microbats update their locations and move toward prey based on the echo sound and the location of other microbats, determined by interacting with each other. Furthermore, the microbats' velocity becomes higher and the pulses lower when the microbats are close to the prey location.

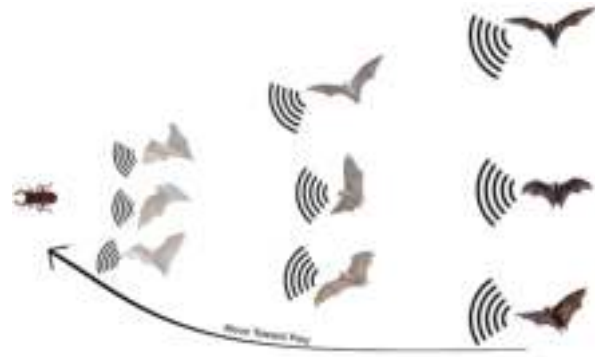


Figure 1. Bat movement toward prey.

In the BAT algorithm, each microbat is considered as a solution in the population, the position of the prey is the optimal solution for a particular optimization problem, the location is presented in a vector x with a range between f_{min} and f_{max} , the velocity of microbats is presented in a vector v , and the wavelength is λ with pulse rate $r \in [0, 1]$ and loudness range A , where the largest is presented as A_0 and the lowest is A_{min} .

3.2. Procedure of BAT

This section presents the general optimization procedure of the BAT algorithm. This optimization procedure contains six main steps, which are presented in Figure 2 and thoroughly discussed below:

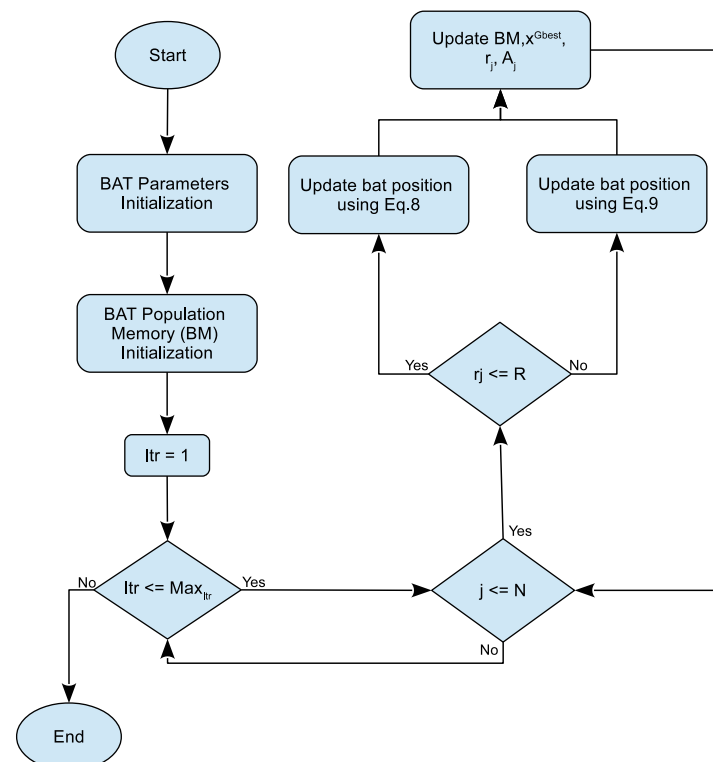


Figure 2. Bat algorithm flowchart.

- Step 1: BAT algorithm parameter initialization. In this step, the BAT parameters are initialized with initial values. These parameters are the number of microbats

(solutions) (N), maximum number of iterations (Max_{Iter}), $\max(f_{max})$ and $\min(f_{min})$ frequency range, velocity vector (v), loudness rate (A), pulse rate (r), initial pulse rate (r^0), two parameters (λ and α) with constant values in the range $[0, 1]$, and bandwidth range (ϵ) in the range $[-1, 1]$.

- Step 2: BAT population memory initialization. The BAT solutions are randomly generated in this step using Equation (1), considering the BAT algorithm and particular optimization problem parameters and constraints.

$$x_i^j = lb_i + (ub_i - lb_i) \times R, \quad (1)$$

where $\forall i = 1, 2, \dots, d, \forall j = 1, 2, \dots, N$, and R is a random number between 0 and 1. The produced solutions together generate the population, as shown in Equation (2), and are stored in the BAT memory (BM) in ascending order on the basis of their fitness values. The best solution with the fittest values is assigned to x^{Gbest} and stored in the first position of the BM.

$$BM = \begin{bmatrix} x_1^1 & x_2^1 & \cdots & x_d^1 \\ x_1^2 & x_2^2 & \cdots & x_d^2 \\ \vdots & \vdots & \cdots & \vdots \\ x_1^N & x_2^N & \cdots & x_d^N \end{bmatrix}, \quad (2)$$

- Step 3: Bat population intensification. Now, all microbats x fly and change their position considering a velocity v defined by a frequency f that is generated randomly, as shown in Equations (3) and (4). Accordingly, the bat positions are updated using Equation (5).

$$f_i = f_{min} + (f_{min} - f_{max}) \times R, \quad (3)$$

$$v_i'^j = v_i^j + (x_i^j - x^{Gbest}) \times f_i, \quad (4)$$

$$x_i'^j = x_i^j + v_i'^j, \quad (5)$$

It is notable that the microbats update their locations to be closer to the global best (x^{Gbest}). Thus, the new position of the microbats intensifies the position in a direction toward x^{Gbest} .

- Step 4: Bat population diversification. In this step, the microbats' positions are updated based on random parameters to attempt to find better global solutions. A new solution is selected from the BM based on a selection method and assigned to x^{Gbest} . Subsequently, the current solution is updated using x^{Gbest} as follows:

$$x_i'^j = x^{Gbest} + \epsilon \times A_j', \quad (6)$$

where A' is the loudness average. As Steps 3 and 4 are contradictory steps, the BAT algorithm chooses one of them at a time for execution. This selection is based on the following equation:

$$x'^j = \begin{cases} x^{Gbest} + \epsilon \times A_j' & \text{if } r_j \leq R \\ x^j + v'^j & \text{otherwise} \end{cases}, \quad (7)$$

- Step 5: BM update. The position of the microbats in the BM will be updated in this step if the new location is fitter than the old one and $R \leq A_j$. Moreover, x^{Gbest} will be

updated if the new BM contains a solution with a better fitness value. Subsequently, r_j and A_j values will be updated in accordance with Equations (8) and (9).

$$r_j = r_j^0(1 - e^{(-\gamma \times itr)}), \quad (8)$$

$$A_j = \alpha \times A_j, \quad (9)$$

where itr denotes the current iteration number.

- Step 6: Stop criterion. Steps 3, 4, and 5 will be repeated until the algorithm reaches the stop criterion.

4. Proposed Method

This section presents the main contribution of this paper. The proposed method includes five phases, which are presented in Figure 3 and thoroughly discussed below.

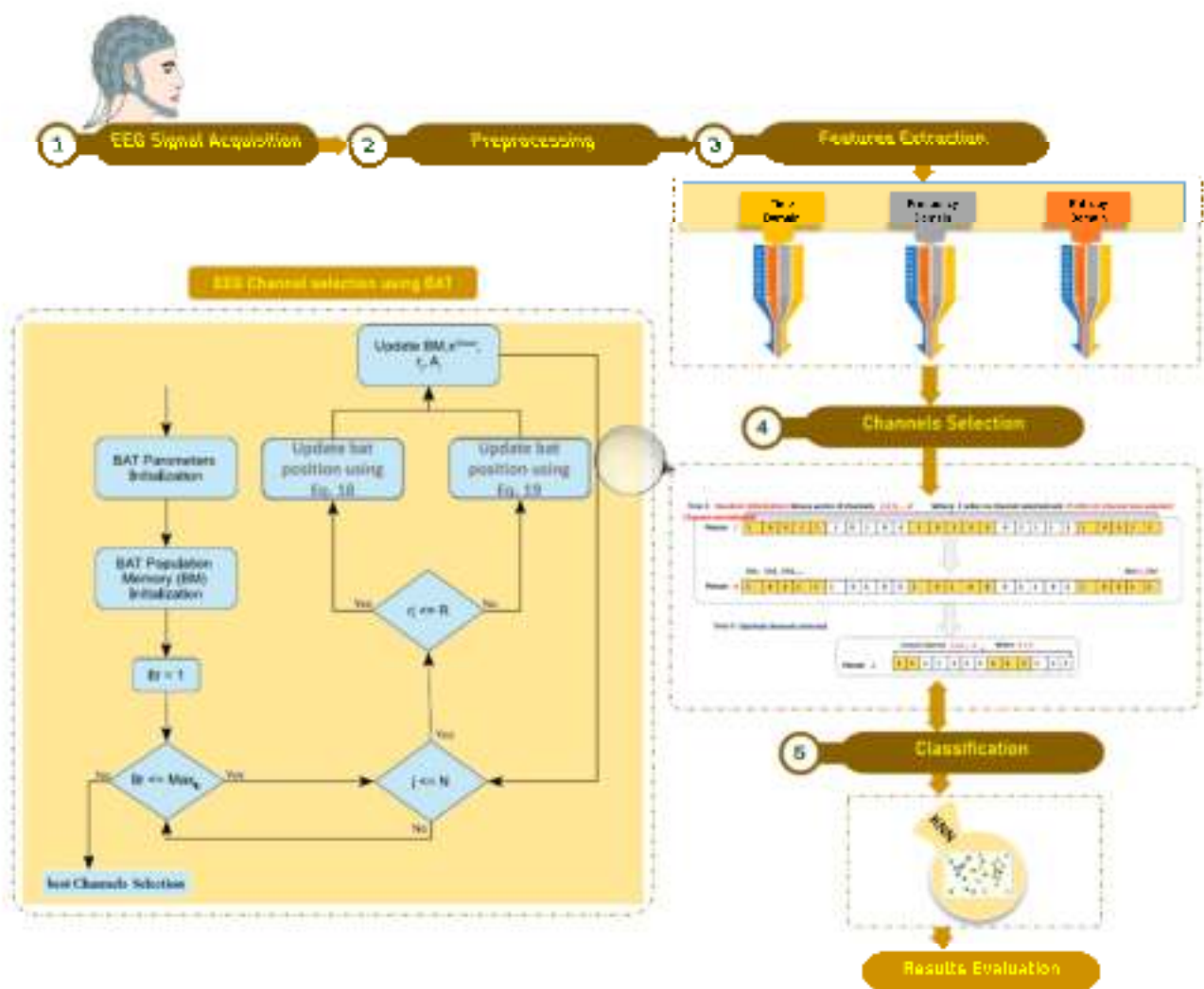


Figure 3. A proposed method for electroencephalogram channel selection.

4.1. Phase I: EEG Signal Acquisition

In this study, the source of the dataset is from g.tec Medical Engineering GmbH, a standard EEG dataset; details about this dataset are explained in [8]. The EEG data of poststroke patients with hemiparesis of the upper extremities were studied. This study involved 8 poststroke patients treated with the recoveriX system (g.tec medical engineering

GmbH), with a mean age of 22 years ($SD = 4.582$). Each participant received BCI-based MI training for three months, with 2 training sessions each week (for a total of 25 training sessions). The study team conducted and analyzed two assessments (pre- and post-training). The pretraining evaluation was scheduled 30 to 35 days prior to the intervention, and the post-training evaluation was conducted a few days after the intervention (see Figure 4). The Ethikkommission des Landes Oberösterreich in Austria (#D-42-17) authorized this study protocol, and each patient signed an informed consent form prior to the preassessment. Finally, this dataset is captured with a sample rate of 256 Hz.

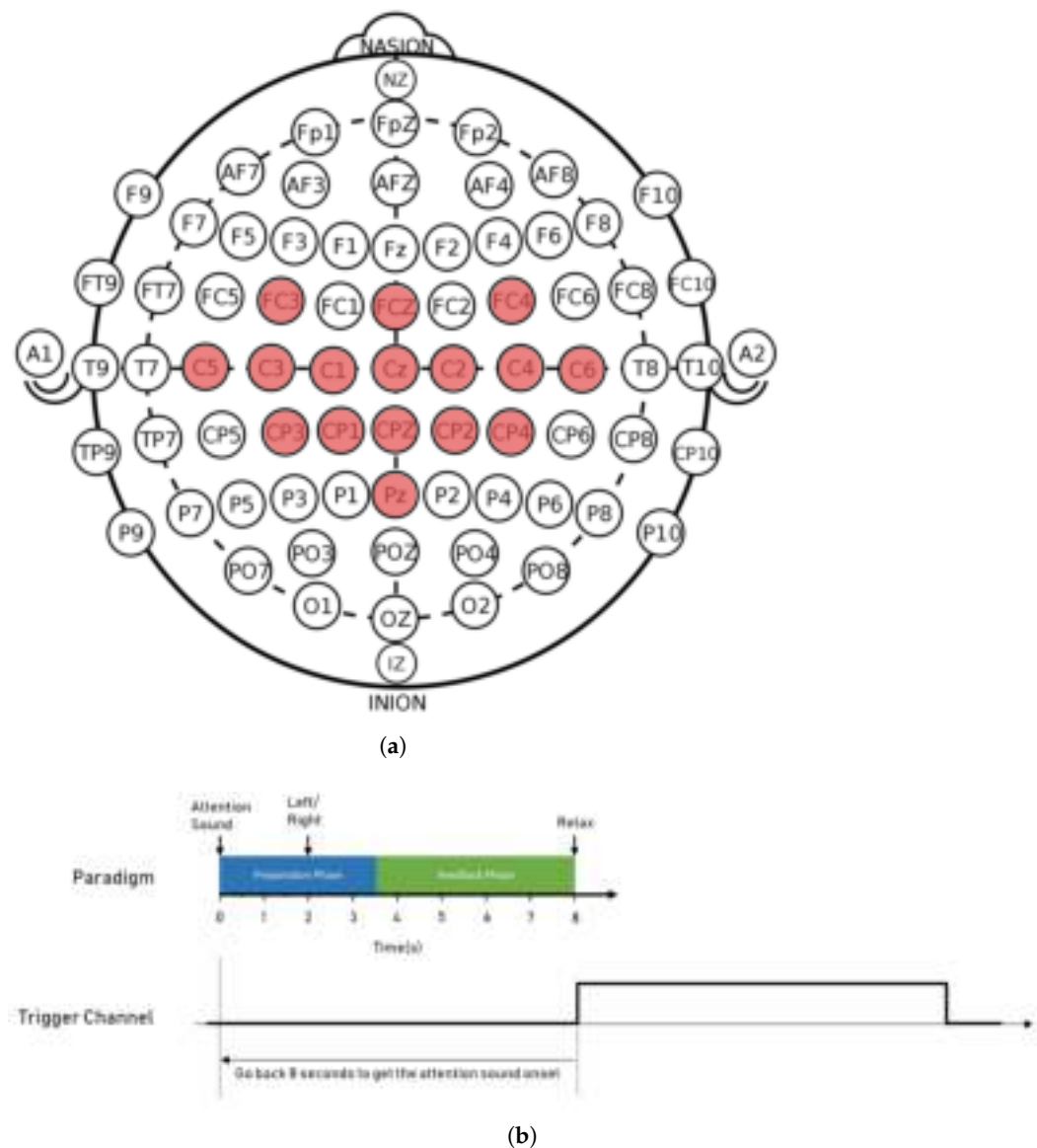


Figure 4. (a) EEG electrode distributions based on 10–20 system; (b) schematic diagram of EEG recording protocol.

According to system indications for an MI mental task, the patients were instructed to visualize dorsal wrist movement. Prior to and after EEG neurorehabilitation, the patients participated in 25 MI-based BCI sessions with follow-up evaluation visits to measure the functional changes. Each session consisted of 240 MI repetitions with both hands, broken down into three 80-trial runs. Each session lasted approximately one hour, including the time required for setup and cleanup. The MI-based BCI tasks were illustrated with randomized inter-trial intervals in a pseudo-random order.

For capturing all the various neural processes that take place during stroke rehabilitation—involving cognitive, affective, and motor components—sixteen-channel active EEG caps were employed. The caps were made by the g.tec medical engineering GmbH company in Austria.

According to the international 10–20 system, the EEG electrodes were positioned as follows: (FC5, FC1, FCz, FC2, FC6, C5, C3, C1, Cz, C2, C4, C6, CP5, CP1, CP2, and CP6). To the subject, a ground electrode was applied at FPz and a reference electrode was affixed at the right earlobe.

These channels are linked with different cognitive and sensorimotor processes which are in some way impaired in patients after a stroke. Particularly, the frontocentral channels (FC5, FC1, FCz, FC2, FC6) are associated with emotional regulation, working memory, and secondary motor area. Further, it is important to note that the central channels (C5, C3, C1, Cz, C2, C4, and C6) are related to sensory–motor integration and body movement representation. In addition, the centroparietal channels (CP5, CP1, CP2, CP6) are described to be involved in the process of integration of the exteroceptive and the proprioceptive inputs as well as in the regulation of the voluntary movements [22].

4.2. Phase II: Preprocessing

Each channel of the recorded EEG dataset initially used two standard filters. First, a bandpass filter (BPF) with frequencies between (8 and 30 Hz) was used to confine the band of the recorded EEG data, and second, a Butterworth (BW) notch filter at (50 Hz) was used to reduce the noise from power line interference. (ICs) $s(t) = [s_1(t), \dots, s_n(t)]$, utilizing the *FastICA* algorithm proposed by [55]. The set $s(t)$ of n unknown components that were linearly mixed within matrix A and the set $x(t)$ of n observations where $x(t) = [x_1(t), \dots, x_n(t)]$ [56–58] represent the EEGs and are related to $s(t)$, t , which is the time; hence, the ICA equation is

$$x(t) = As(t) \quad (10)$$

Then, three metrics were used to evaluate the artifactual components, ICs: kurtosis (*Kurt*), skewness (*Skw*), and sample entropy (*SampEn*). If these parameters surpassed the $pm1.2$ threshold for each IC, the IC was marked as critical and denoised using WT. The practical value of the threshold was determined through trial and error and previous research [59–61]. The threshold value of ± 1.2 is not a drawback of the AICA–WT technique, as the artifactual ICs were not rejected but were denoised using the WT technique. As a result, WT denoised the marked ICs before returning the enhanced components to the original EEG dataset [22,23].

4.3. Phase III: EEG Feature Extraction

Three different feature extraction techniques are used in this work. These techniques include time domain features, entropy features, and frequency domain features as extracted in [8].

4.4. Phase IV: EEG Channel Selection Using BAT Optimizer

This phase is the main contribution, and it includes several steps to achieve the optimal subset of the EEG channels which can provide the highest accuracy rate. The following steps represent how we adapt the BAT optimizer for the EEG channel selection problem:

- Step 1: BAT algorithm parameter initialization. In this step, the BAT parameters are initialized with initial values. These parameters are the number of microbats (solutions) (N), maximum number of iterations (Max_{Itr}), max (f_{max}) and min (f_{min}) frequency range, velocity vector (v), loudness rate (A), pulse rate (r), initial pulse rate (r^0), two parameters (λ and α) with constant values in the range $[0, 1]$, and bandwidth range (ϵ) in the range $[-1, 1]$, as shown in Table 1.

Table 1. Metaheuristic parameters.

Algorithm	Parameters	Population Size	Runs
CS	$\beta = 1.5, P_a = 0.25$	20	30
PSO	$c1 = c2 = 2$	20	30
BAT	$A = 0.5, r = 0.5$	20	30
FFA	$\alpha = 0.5, \gamma = 1, \beta_0 = 0.2$	20	30
GWO	-	20	30

- Step 2: BAT population memory initialization. The BAT solutions are randomly generated in this step using Equation (11), considering the BAT algorithm and particular optimization problem parameters and constraints.

$$x_i^j = lb_i + (ub_i - lb_i) \times R, \quad (11)$$

where $\forall i = 1, 2, \dots, d, \forall j = 1, 2, \dots, N$, and R is a random number between 0 and 1. The produced solutions together generate the population, as shown in Equation (12), and are stored in the BAT memory (BM) in ascending order on the basis of their fitness values. The best solution with the fittest values is assigned to x^{Gbest} and stored in the first position of the BM.

$$BM = \begin{bmatrix} x_1^1 & x_2^1 & \cdots & x_d^1 \\ x_1^2 & x_2^2 & \cdots & x_d^2 \\ \vdots & \vdots & \cdots & \vdots \\ x_1^N & x_2^N & \cdots & x_d^N \end{bmatrix}, \quad (12)$$

- Step 3: BAT population intensification. Now, all microbats x fly and change their position considering a velocity v defined by a frequency f that is generated randomly, as shown in Equations (13) and (14). Accordingly, the bat positions are updated using Equation (15).

$$f_i = f_{min} + (f_{max} - f_{min}) \times R, \quad (13)$$

$$v_i^j = v_i^j + (x_i^j - x_i^{Gbest}) \times f_i, \quad (14)$$

$$x_i'^j = x_i^j + v_i^j, \quad (15)$$

It is notable that the microbats update their locations to be closer to the global best (x^{Gbest}). Thus, the new position of the microbats intensifies the position in a direction towards x^{Gbest} .

- Step 4: BAT population diversification. In this step, the microbats' positions are updated based on random parameters to attempt to find better global solutions. A new solution is selected from the BM based on a selection method and assigned to x^{best} . Subsequently, the current solution is updated using x^{best} , as follows:

$$x_i'^j = x^{best} + \varepsilon \times A'_j, \quad (16)$$

where A' is the loudness average. As Steps 3 and 4 are contradictory steps, the BAT algorithm chooses one of them at a time for execution. This selection is based on the following equation:

$$x'^j = \begin{cases} x^{best} + \varepsilon \times A'_j & \text{if } r_j \leq R \\ x^j + v'^j & \text{otherwise} \end{cases}, \quad (17)$$

- Step 5: BM update. The position of the microbats in the BM will be updated in this step if the new location is fitter than the old one and $R \leq A_j$. Moreover, X^{Gbest} will be updated if the new BM contains a solution with a better fitness value. Subsequently, r_j and A_j values will be updated in accordance with Equations (18) and (19).

$$r_j = r_j^0(1 - e^{(-\gamma \times itr)}), \quad (18)$$

$$A_j = \alpha \times A_j, \quad (19)$$

where itr denotes the current iteration number.

- Step 6: stop criterion. Steps 3, 4, and 5 will be repeated until the algorithm reach the stop criterion.

4.5. Phase V: Classification

A KNN classifier is used in this work. In this proposed method, the dataset is split into 60% training, 20% validation, and 20% testing.

5. Experiments and Results

This section thoroughly explains the performance of the proposed method (BAT) of EEG channels for stroke patient rehabilitation. Since the proposed approach is non-deterministic, the mean accuracy rate over 20 runs is computed to avoid biased results. The experiments use a Ideapad 310 Lenovo PC, China. Intel Core i7[®] 2.59 Ghz processor, 16 GB of RAM, and official Windows 10. The performance of the proposed method is evaluated using two measures, namely, accuracy and the number of EEG channels selected, with light shed on the stability of the algorithms by computing the best, worst, and mean accuracy. Moreover, a statistical test is adopted based on the classification accuracy of the motor/imaging EEG dataset using the sum of ranks of the metaheuristic algorithms. The proposed method is tested using three types of EEG features: time domain features, frequency domain features, and entropy domain features. Furthermore, the performance results of the proposed method are evaluated using a statistical test to show the significance of the proposed method compared with other metaheuristic algorithms.

Table 1 shows the parameters used for the selected metaheuristic algorithms that are used in this work.

5.1. Time Domain Results

Table 2 presents the results of seven features in the time domain: Higuchi's fractal dimension (HFD); the Hjorth parameters Hjorth activity (HjAc), Hjorth complexity (HjComp), and Hjorth mobility (HjMo); Hurst (Hur); kurtosis (Kurt); and skewness (Skw). The accuracy and the number of EEG channels selected are two measurements to evaluate the seven features chosen for the motor/imaging EEG dataset. The evaluation metrics are classified into three categories: best, worst, and mean accuracy and no. of channels selected to evaluate the algorithms' stability.

According to Table 2, the performance results for the metaheuristic algorithms PSO, GWO, FFA, CS, and BAT with several features achieved different and remarkable accuracy concerning best, worst, and mean accuracy and the number of channels selected. For the HFD feature, the BAT algorithm achieved better accuracies of 99, 97.5, and 92.8 for best, mean, and worst accuracy and channel selection, respectively. For the HjAc feature, the BAT algorithm achieved high accuracy, 97.7, for accuracy and no. of channels selected in the best case, while PSO achieved a valuable ratio of 94.4 and 85.3 in accuracy and the number of channels selected in the mean and worst cases, respectively. For the HjComp feature, the PSO algorithm achieved an accuracy of 88.6 and 84.7 concerning mean and worst cases, while the BAT algorithm achieved the highest accuracy of 93.8 in the best case. For the HjMo feature, BAT obtained 97.8, 95.4, and 88.6 for best, mean, and worst among other metaheuristic algorithms. For the Hur feature, PSO gained high accuracies of 72.8, 65.2, and 56.6 for best, mean, and worst, respectively. For the Kurt feature, the BAT

algorithm obtained valuable accuracies of 73.2, 68.8, and 63.1 for best, mean, and worst accuracy and the number of channels selected, respectively.

Moreover, for Skw, BAT achieved high accuracies of 79.2 and 72.5 for best and mean, respectively, compared with PSO, which obtained an accuracy of 58 in the worst case of accuracy. Figure 4 shows the results of the time domain features for the convergence rate and channel distribution, comparing the BAT and PSO algorithms, which achieved the highest accuracy over the other metaheuristic algorithms, i.e., GWO, FFA, and CS. In Figure 4a, the convergence rate shows the channels and the number of iterations for the BAT and PSO algorithms. In addition, Figure 4b presents the effective and emotive channels when channels are distributed in the BAT and PSO algorithms.

Table 2. The performance of metaHeuristic algorithms with several feature extraction approaches for motor/imaging EEG dataset.

Feature/Algorithm			PSO	GWO	FFA	CS	BAT
HFD	Best	Accuracy	97.9	90.7	66.7	54.9	99
		No. channels	8	3	1	1	6
	Worst	Accuracy	81.5	67.4	45.1	43.2	92.8
		No. channels	12	6	3	2	13
	Mean	Accuracy	94.4	76.2	56.7	50.6	97.5
		No. channels	9	4	2	2	10
HjAc	Best	Accuracy	97.1	88.4	68.8	55.1	97.7
		No. channels	5	3	1	1	4
	Worst	Accuracy	85.3	70.4	41.7	42.1	77.9
		No. channels	9	5	3	2	14
	Mean	Accuracy	94.4	82.1	60.7	48.7	90.8
		No. channels	8	4	3	2	8
HjComp	Best	Accuracy	91.2	78.9	58.8	48.7	93.8
		No. channels	7	3	2	1	5
	Worst	Accuracy	84.7	58.1	47.4	40.9	74.7
		No. channels	15	6	3	2	13
	Mean	Accuracy	88.6	69.9	53.9	44.2	88.2
		No. channels	9	4	3	1	9
HjMo	Best	Accuracy	96.5	89.3	67.9	53.5	97.9
		No. channels	6	2	2	1	5
	Worst	Accuracy	86.4	51.8	53.9	43.8	88.6
		No. channels	11	6	3	2	12
	Mean	Accuracy	92.3	80.6	61.2	47.9	95.4
		No. channels	9	4	3	1	9
Hur	Best	Accuracy	72.8	57.9	47.2	44.1	70.1
		No. channels	4	2	1	1	5
	Worst	Accuracy	56.6	44.1	41.5	41.3	53.9
		No. channels	10	6	3	2	11
	Mean	Accuracy	65.2	51.7	44.7	42.6	64.9
		No. channels	8	4	3	2	9

Table 2. Cont.

Feature/Algorithm			PSO	GWO	FFA	CS	BAT
Kurt	Best	Accuracy	71.2	61	48.4	43.4	73.2
		No. channels	7	3	2	1	7
	Worst	Accuracy	56.5	48.2	43.3	40.3	63.1
		No. channels	12	6	3	2	12
	Mean	Accuracy	65.6	54.9	45.4	42	68.8
		No. channels	9	5	3	2	10
Skw	Best	Accuracy	78.7	63.7	50	44.4	79.2
		No. channels	4	1	2	1	3
	Worst	Accuracy	58	41	43.2	40.4	48.2
		No. channels	11	6	3	2	13
	Mean	Accuracy	69.6	53.3	45.9	42.3	72.5
		No. channels	7	4	2	2	9

5.2. Entropy Domain Results

Table 3 shows the results of the performance accuracy using seven entropy features, namely, ConFuzEn, FuzEn, impe, MFEmu, RCMFEmu, SampEn, and TsEn. Based on the accuracy and the number of selected channels for best, worst, and mean cases, these entropy features are used to compare the five metaheuristic algorithms, PSO, GWO, FFA, CS, and BAT, respectively. Compared to the other seven entropy features, the best classification performance achieved the highest accuracy using ConFuzEn with the PSO algorithm compared to the GWO, FFA, CS, and BAT algorithms. Other entropy features produced higher accuracy results with the BAT algorithm than the PSO, GWO, FFA, and CS algorithms. The BAT algorithm registered accuracies of 95.5, 95.6, 88.1, 87.2, 91.4, and 84.6 using FuzEn, impe, MFEmu, RCMFEmu, SampEn, and TsEn, respectively. Furthermore, Figure 5 presents the performance evaluation results between the BAT and PSO algorithms concerning the convergence rate and channel distribution.

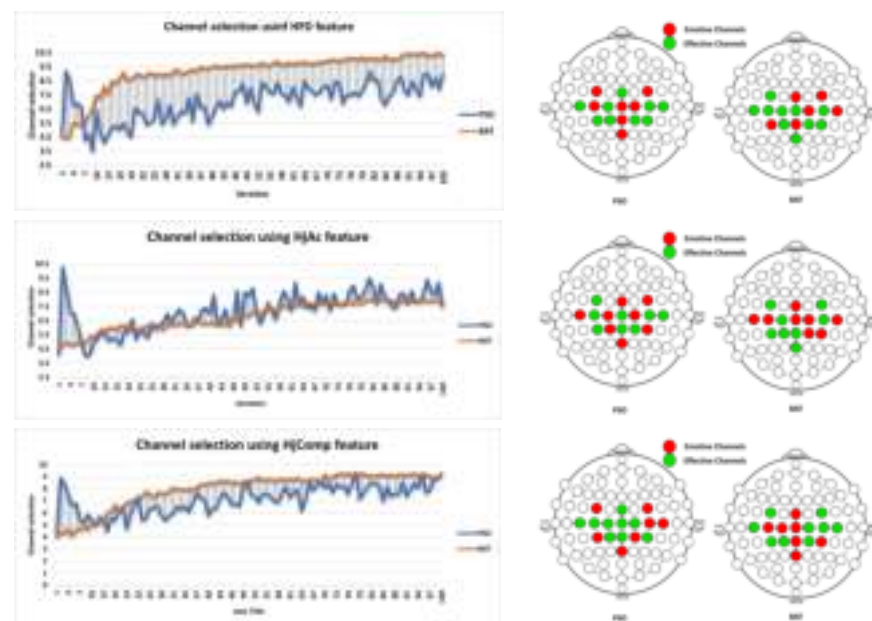


Figure 5. Cont.

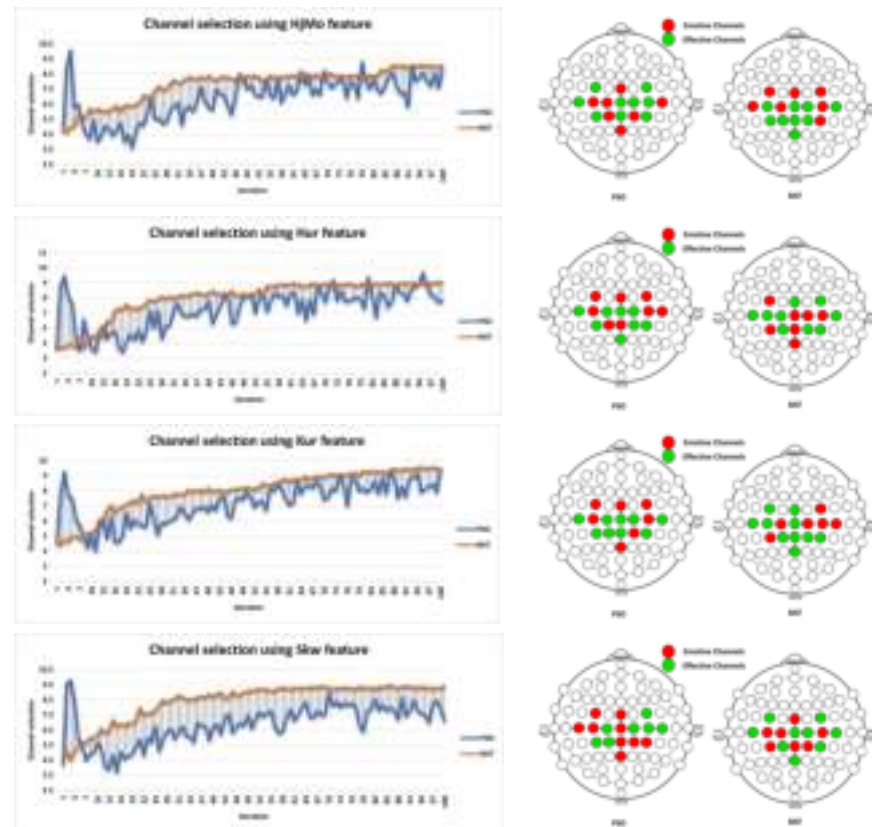


Figure 5. Convergence rate and channel distribution.

Table 3. The performance of metaHeuristic algorithms with several feature extraction approaches for motor/imaging EEG dataset.

Feature/Algorithm			PSO	GWO	FFA	CS	BAT
ConvFuzEn	Best	Accuracy	96.6	82.7	61.3	52	96.5
		No. channels	3	3	1	1	7
	Worst	Accuracy	76.4	64.7	45.5	42.5	87.1
		No. channels	11	6	3	2	14
	Mean	Accuracy	90.2	73.6	54.9	45.8	93.2
		No. channels	8	5	2	1	11
FuzEn	Best	Accuracy	93.6	81.6	60.6	49.1	95.5
		No. channels	2	4	2	1	8
	Worst	Accuracy	82.2	68.7	50.5	41.8	88.9
		No. channels	10	6	3	2	12
	Mean	Accuracy	88.6	74.4	55.7	45.5	92.1
		No. channels	7	5	3	1	10
impen	Best	Accuracy	94.7	78.6	59.3	49.8	95.6
		No. channels	5	3	2	1	7
	Worst	Accuracy	75.8	56.1	47.9	44.3	84.9
		No. channels	10	6	3	2	12
	Mean	Accuracy	85.3	69.8	52.3	46.6	90.8
		No. channels	8	5	2	2	10

Table 3. Cont.

Feature/Algorithm			PSO	GWO	FFA	CS	BAT
MFEmu	Best	Accuracy	85.2	73.6	54.3	46.8	88.1
		No. channels	7	3	2	1	7
	Worst	Accuracy	70.8	55.9	45.1	40.5	75.5
		No. channels	12	6	3	2	12
	Mean	Accuracy	80.7	66.1	49.4	44.8	84
		No. channels	9	5	2	2	9
RCMFEmu	Best	Accuracy	84.3	67.7	51.9	46.6	87.2
		No. channels	4	2	2	1	8
	Worst	Accuracy	63.7	47	44.9	41.2	75.3
		No. channels	12	6	3	2	13
	Mean	Accuracy	77.3	56.5	47.7	43	80.3
		No. channels	8	4	2	1	10
SampEn	Best	Accuracy	87.9	71.5	52.9	51.9	91.4
		No. channels	6	3	1	1	7
	Worst	Accuracy	75.4	53.2	42.6	40.4	75.9
		No. channels	10	6	3	3	14
	Mean	Accuracy	81.4	63.7	48.2	44.3	83.6
		No. channels	8	5	2	2	10
TsEn	Best	Accuracy	83.9	72.5	52.3	46.6	84.6
		No. channels	7	3	1	1	4
	Worst	Accuracy	65.9	54.4	41.2	40.2	61.3
		No. channels	12	7	3	2	13
	Mean	Accuracy	78.7	64.5	47.2	42.8	79.8
		No. channels	10	5	2	1	10

5.3. Frequency Domain Results

Based on Table 4, the frequency domain features of meanF and medF obtained high accuracies of 96.7 and 83.4 for the PSO and BAT algorithms, respectively. In the best, worst, and mean cases, meanF gained a significant accuracy relative to the medF feature on the overall accuracy results for the PSO and BAT algorithms. The best accuracy of medF was obtained at 96.7 with PSO, while the worst and mean achieved accuracies were 90.3 and 93.8 with the BAT algorithm, respectively. Moreover, the BAT algorithm scored high accuracy classification results of 83.4 and 78.4 relative to the best and worst accuracies with the feature of medF, respectively, while PSO gained an accuracy of 69.5. The convergence rate and the channel distribution are shown in Figure 6.

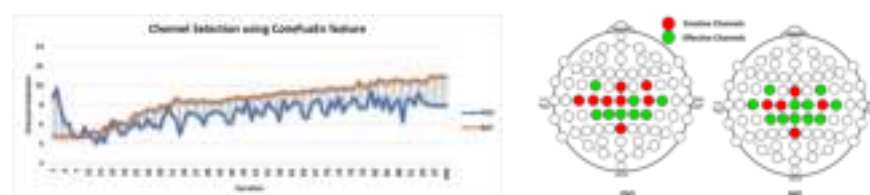


Figure 6. Cont.

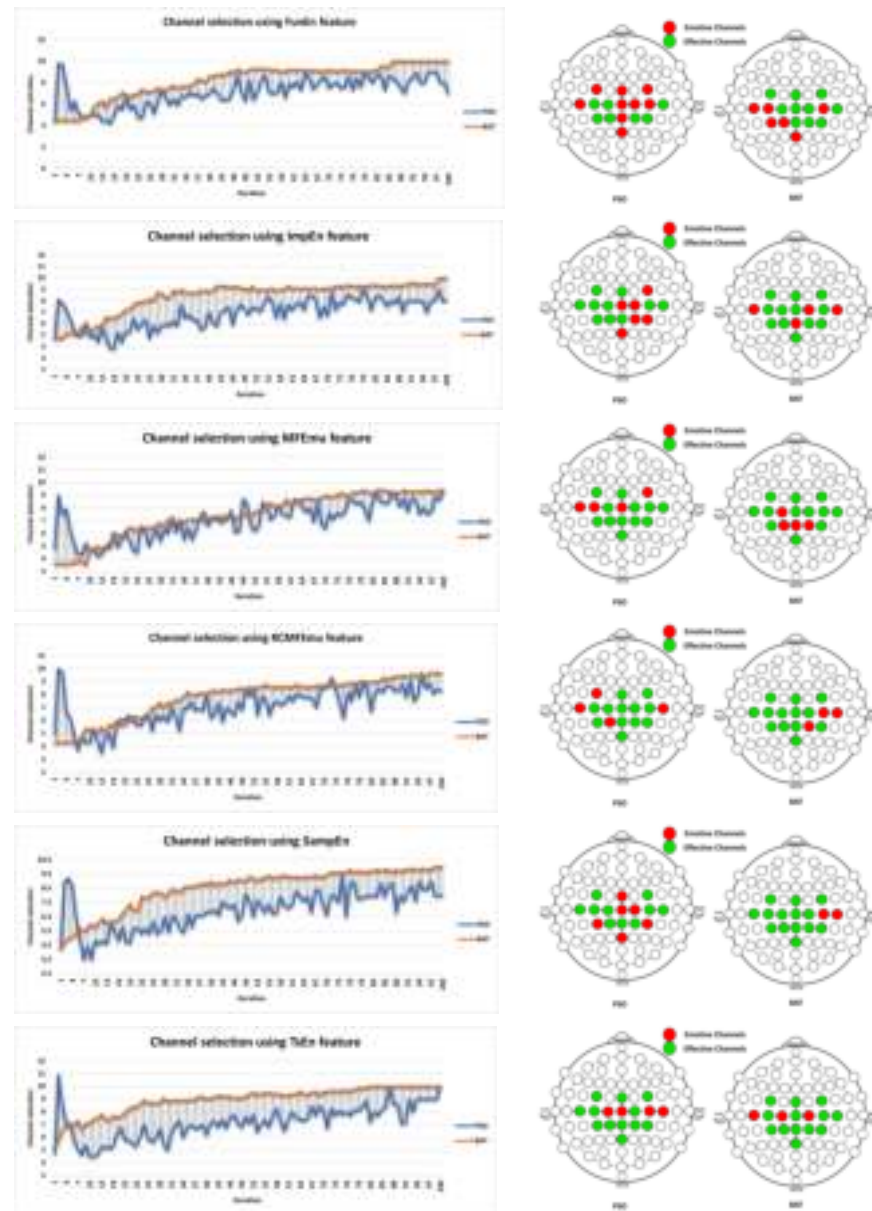


Figure 6. Convergence rate and channel distribution.

As mentioned, Table 4 shows the performance of the proposed method (BAT) compared with the PSO, GWO, FFA, and CS algorithms using frequency domain features. The BAT algorithm achieved better results than all other algorithms except in some cases in which the PSO algorithm slightly outperformed it, such as in *meanF*, where the best accuracy was 96.7 for PSO and 96.3 for BAT. However, the overall runs of the BAT algorithm achieved better results with accuracy measures equal to 93.8, 91.2, 76.4, 54, and 50 for the BAT, PSO, GWO, FFA, and CS algorithms, respectively. The convergence rate and the channel distribution are shown in Figure 7.

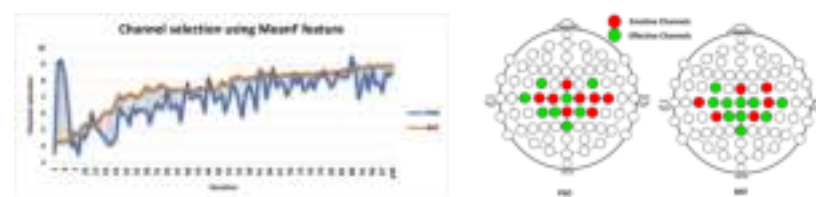


Figure 7. Cont.

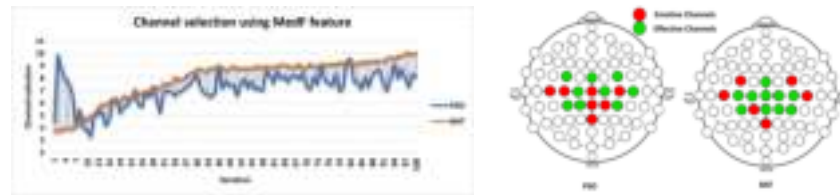


Figure 7. Convergence rate and channel distribution.

Table 4. The performance of BAT algorithm compared with several metaheuristic algorithms using frequency domain features.

Feature/Algorithm			PSO	GWO	FFA	CS	BAT
<i>meanF</i>	Best	Accuracy	96.7	87.3	64.1	52.5	96.3
		No. channels	5	3	2	1	7
	Worst	Accuracy	65.4	66	50.2	41.9	90.3
		No. channels	10	6	3	2	12
	Mean	Accuracy	91.2	76.4	54	50	93.8
		No. channels	8	4	2	2	10
<i>MedF</i>	Best	Accuracy	82.7	69.2	53.6	46.6	83.4
		No. channels	6	3	2	1	6
	Worst	Accuracy	69.5	55.5	41	20	68.7
		No. channels	10	6	3	2	14
	Mean	Accuracy	76.3	60.6	46.1	36.5	78.4
		No. channels	9	4	2	2	10

5.4. Statistical Analysis Results

Before presenting our findings for the accuracy and no. of channels selected for improving the classification performance of the stroke EEG signals' time, entropy, and frequency features analysis, we would like to examine the influence of classification accuracy in the motor/imaging EEG dataset by the adoption of the sum of ranks of the metaheuristic algorithms. In this way, the feature that improved the rank accuracy of selected channels can be identified.

As shown in Table 5, we performed the sum of ranks of the PSO, GWO, FFA, CS, and BAT metaheuristic algorithms according to the individual feature domain. In the time domain, we tested seven features. The BAT algorithm significantly improved the accuracy rank, especially in HFD and Hjac, for an accuracy rank of 99 and 97.7, respectively. Moreover, other features in the time domain, such as HjComp, HjMo, Kurt, and Skw, showed an influential impact on the classification accuracy results relative to the sum of the rank of the BAT algorithm except for the Hur feature, which had a high effect on the PSO algorithm. However, in the statistical test phase, the BAT algorithm had low summation-of-rank results among other metaheuristic algorithms.

On the other hand, the sum of ranks for the BAT algorithm showed a meaningful impact on the classification accuracy results relative to the seven features of the entropy domain except for the ConvFuzEn feature, which had a remarkable effect on the PSO algorithm with a difference of 0.1 compared with BAT concerning the sum-of-rank results in the statistical analysis. However, the BAT algorithm had low summation-of-rank results in the statistical test phase among the other metaheuristic algorithms.

However, in the classification accuracy results from the frequency domain features, meanF showed a salient ameliorating effect on the accuracy with the PSO algorithm relative to the accuracy rank 96.7 compared with the medF feature, which had an 83.4 accuracy

rank with the BAT algorithm. Moreover, the BAT and PSO algorithms had low summation-of-rank results among other metaheuristic algorithms.

Table 5. The sum of ranks of metaHeuristic algorithms based on classification accuracy on motor/imaging EEG dataset.

Feature/Algorithm			PSO	GWO	FFA	CS	BAT
TDF	HFD	Accuracy	97.9	90.7	66.7	54.9	99
		Rank	2	3	4	5	1
	HjAc	Accuracy	97.1	88.4	68.8	55.1	97.7
		Rank	2	3	4	5	1
	HjComp	Accuracy	91.2	78.9	58.8	48.7	93.8
		Rank	2	3	4	5	1
	HjMo	Accuracy	96.5	89.3	67.9	53.5	97.9
		Rank	2	3	4	5	1
	Hur	Accuracy	72.8	57.9	47.2	44.1	70.1
		Rank	1	3	4	5	2
	Kur	Accuracy	71.2	61	48.4	43.4	73.2
		Rank	2	3	4	5	1
	Skw	Accuracy	78.7	63.7	50	44.4	79.2
		Rank	2	3	4	5	1
	Summation of Ranks			13	21	28	35
FDF	meanF	Accuracy	96.7	87.3	64.1	52.5	96.3
		Rank	1	3	4	5	2
	MedF	Accuracy	82.7	69.2	53.6	46.6	83.4
		Rank	2	3	4	5	1
Summation of Ranks			3	6	8	10	3
EDF	ConvFuzEn	Accuracy	96.6	82.7	61.3	52	96.5
		Rank	1	3	4	5	2
	FuzEn	Accuracy	93.6	81.6	60.6	49.1	95.5
		Rank	2	3	4	5	1
	impen	Accuracy	94.7	78.6	59.3	49.8	95.6
		Rank	2	3	4	5	1
	MFEmu	Accuracy	85.2	73.6	54.3	46.8	88.1
		Rank	2	3	4	5	1
	RCMFEmu	Accuracy	84.3	67.7	51.9	46.6	87.2
		Rank	2	3	4	5	1
	SampEn	Accuracy	87.9	71.5	52.9	51.9	91.4
		Rank	2	3	4	5	1
	TsEn	Accuracy	83.9	72.5	52.3	46.6	84.6
		Rank	2	3	4	5	1
	Summation of Ranks			13	21	28	35

Figure 8 shows the results of the speed of metaheuristic algorithms in seconds.

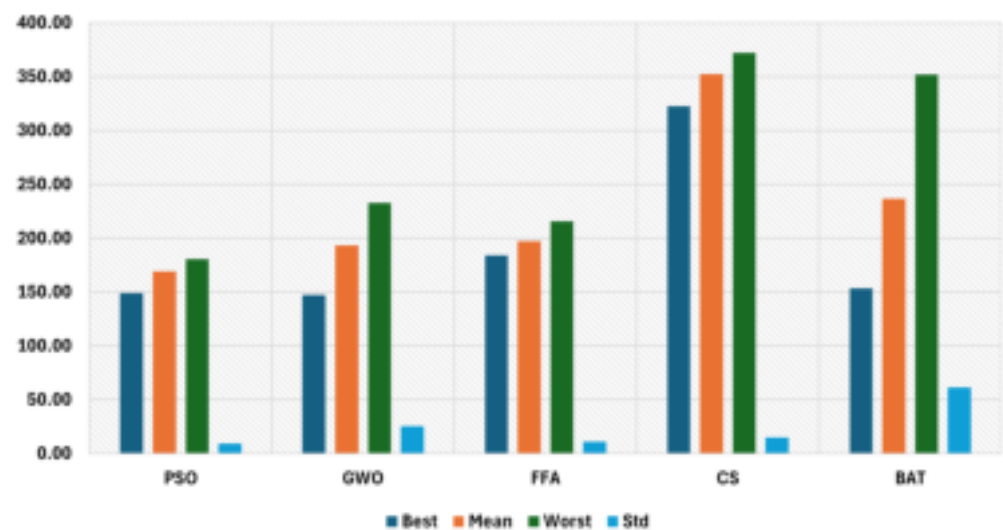


Figure 8. Speed of metaheuristic algorithms in seconds.

5.5. Results Discussion

This research aims to choose the most relevant EEG channels that produce an effective accuracy rate for stroke patient rehabilitation by using a binary bat algorithm-based optimization technique to optimize TEF characteristics. To further improve the classification performance of stroke EEG signals, three types of features (namely, time, entropy, and frequency domain features) were used as the baseline for the classification task. Two measurements were applied to evaluate the performance of the proposed method, accuracy and no. of EEG channels selected. Furthermore, we adopted a statistical test using the sum of the ranks for the motor/imaging EEG dataset to examine the influence of the classification accuracy and the significance of the suggested method compared with state-of-the-art metaheuristic algorithms such as PSO, GWO, FFA, and CS. Firstly, the accuracy and the no. of EEG channels selected were evaluated by computing the best, worst, and mean accuracy values to capture the stability of the implemented algorithm. In the time domain, seven features were tested, and the HFD feature had the most significant influence on the improvement of the classification task in the BAT optimizer among other optimization algorithms for the best, mean, and worst cases of accuracy and the no. of channels selected, respectively, based on a kNN classifier. In addition, the BAT results outperformed in the testing values for the best accuracy case compared with PSO, which had the highest results compared with the GWO, FFA, and CS algorithms in the HjAc and HjComp features. Moreover, the results for the BAT optimizer registered high best, mean, and worst accuracy values and the no. of channels selected, respectively, with the HFD, HjMo, and Kurt features, respectively, while PSO had the most significant influence on the improvement of the classification task compared with BAT with the Hur feature based on best, mean, and worst accuracy and the no. of channels selected, respectively. For the best-case values, the BAT optimizer outperformed in the valuable and highest values among PSO and the other metaheuristic algorithms with the six features of HFD, HjAc, HjComp, HjMo, Kurt, and Skw, respectively (see Table 2). Furthermore, according to the results of the performance accuracy using seven entropy features based on the KNN classifier, the best classification performance was achieved with the highest accuracy using ConFuzEn with the PSO algorithm compared to the GWO, FFA, CS, and BAT algorithms. Other entropy features produced higher accuracy results with the BAT algorithm than the PSO, GWO, FFA, and CS algorithms according to the best, mean, and worst accuracy values and the no. of channels selected, respectively. The BAT algorithm registered best-case accuracy values of 95.5, 95.6, 88.1, 87.2, 91.4, and 84.6 using the FuzEn, impe, MFEmu, RCMFEmu, SampEn, and TsEn features, respectively, while PSO outperformed BAT with accuracy result values for the ConvFuzEn feature for the KNN classifier (see Table 3).

The BAT and PSO algorithms obtained high accuracies of 96.7 and 83.4, respectively, compared with other algorithms in the literature for meanF and medF frequency domain features. The BAT algorithm scored high accuracy classification results of 83.4 and 78.4 relative to the best and worst accuracies with the feature of medF, respectively, while PSO gained an accuracy of 69.5; furthermore, the PSO and BAT algorithms gained significantly in accuracy relative to the medF feature on the overall accuracy results in the best, worst, and mean accuracy cases. Based on the classification results on the basis of the features from the three different domains types, the BAT and PSO algorithms exerted significant impacts on the classification performance compared with the results of the GWO, FFA, and CS optimization algorithms concerning the accuracy and the no. of channels selected, respectively. Hence, they demonstrate a significant improvement in the average accuracy relative to the classification result on the basis of the individual feature domains, as can be seen in Figure 5, which explains the comparison between the convergence rate and channel distribution for the BAT and PSO algorithms.

Second, we examined the influence of classification accuracy on the motor/imaging EEG dataset using the adopted sum of ranks of metaheuristic algorithms. Thus, the feature that improved the rank accuracy of the selected channels can be identified. The BAT algorithm significantly improved the accuracy rank, especially in the HFD and Hjac features, with an accuracy rank of 99 and 97.7, respectively, compared with the PSO, GWO, FFA, and CS algorithms according to the sum-of-ranks results with respect to the time domain. Furthermore, features such as HjComp, HjMo, Kurt, and Skw showed an influential impact on the classification accuracy results relative to the sum of the rank of the BAT algorithm except for the Hur feature, which had a high effect on the PSO algorithm. However, in the statistical test phase, the BAT algorithm had low summation-of-rank results among the other metaheuristic algorithms. Further, the BAT algorithm showed a meaningful impact on the classification accuracy results for the sum of ranks relative to the seven features of the entropy domain except for the ConvFuzEn feature, which had a remarkable effect on the PSO algorithm with a difference of 0.1 compared with BAT concerning the sum-of-rank results in the statistical analysis. However, in the statistical test phase, the BAT algorithm had a low summation-of-rank results among the other metaheuristic algorithms. Moreover, the BAT and PSO algorithms had low summation-of-rank results among the other metaheuristic algorithms in the frequency domain features. The meanF feature showed a salient ameliorating effect on the accuracy with the PSO algorithm relative to the accuracy rank of 96.7 compared with the medF feature, which had an 83.4 accuracy rank with the BAT algorithm.

6. Conclusions and Future Work

This paper proposes a new method for EEG channel selection based on an optimization algorithm called the BAT-inspired algorithm. The main purpose of the proposed algorithm is to select the most relevant EEG channels that can provide a higher accuracy rate for stroke patient rehabilitation.

The proposed method was tested using a standard EEG dataset collected from eight poststroke patients with hemiparesis of the upper extremities. The EEG data were captured using EEG caps with 16 active electrodes from g.Nautilus PRO, g.tec medical engineering GmbH, Austria.

The proposed method used several EEG feature extraction methods such as the time domain, frequency domain, and entropy domain. In addition, the proposed method (BAT) was compared with four metaheuristic algorithms, namely, particle swarm optimization (PSO), grey wolf optimizer (GWO), cuckoo search (CS), and firefly algorithm (FFA). The performance of the proposed method was evaluated using the accuracy rate and the number of channels selected. The proposed method achieved the best results over all the feature extraction methods and showed a significant improvement using a standard statistical analysis test called the summation of ranks test.

Finally, the proposed method succeeded in reducing the number of EEG channels to less than half while maintaining the accuracy rate.

Regarding future works, the proposed method suffers from some problems; for example, the proposed method is unable to generate new solutions that have the ability to increase accuracy (i.e., stuck in local minima). For that, the proposed method will need to be improved to overcome this problem by modifying the mechanism of the BAT algorithm or hybridizing the BAT algorithm with another metaheuristic algorithm to be able to find the optimal solution. Regarding the second problem, the current version of the BAT algorithm focuses on finding the optimal solution that provides the highest accuracy rate and does not consider the number of EEG channels in the objective function. Therefore, in future work, we will work on the proposed multiobjective BAT version using the highest accuracy and the lowest number of EEG channels simultaneously.

Author Contributions: Conceptualization, M.A.A.-B., Z.A.A.A. and N.K.A.-Q.; methodology, M.A.A.-B., Z.A.A.A. and N.K.A.-Q.; software, M.A.A.-B. and Z.A.A.A.; validation, M.A.A.-B. and S.N.M.; investigation, M.A.A.-B. and Z.A.A.A.; resources, M.A.A.-B., Z.A.A.A. and C.G.; writing—original draft, M.A.A.-B., Z.A.A.A., N.K.A.-Q., S.N.M., N.S.A. and C.G.; Visualization, M.A.A.-B., Z.A.A.A., S.N.M., N.S.A. and C.G. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Deanship of Research and Graduate Studies (DRG) at Ajman University, Ajman, UAE (Grant No. 2022-IRG-ENIT-7).

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: Author Christoph Guger was employed by the g.tec medical engineering GmbH. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

- Li, F.; Fan, Y.; Zhang, X.; Wang, C.; Hu, F.; Jia, W.; Hui, H. Multi-Feature Fusion Method Based on EEG Signal and its Application in Stroke Classification. *J. Med. Syst.* **2020**, *44*, 39. [\[CrossRef\]](#)
- Khairunizam, W.; Yean, C.W.; Murugappan, M.; Junoh, A.K.; Razlan, Z.M.; Shahrman, A.; Mustafa, W.A.W.; Ibrahim, Z.; Nurhafizah, S. An Experimental Framework for Assessing Emotions of Stroke Patients using Electroencephalogram (EEG). *J. Phys. Conf. Ser.* **2020**, *1529*, 052072. [\[CrossRef\]](#)
- Carino-Escobar, R.I.; Carrillo-Mora, P.; Valdés-Cristerna, R.; Rodriguez-Barragan, M.A.; Hernandez-Arenas, C.; Quinzaños-Fresnedo, J.; Galicia-Alvarado, M.A.; Cantillo-Negrete, J. Longitudinal analysis of stroke patients' brain rhythms during an intervention with a brain-computer interface. *Neural Plast.* **2019**, *2019*. [\[CrossRef\]](#) [\[PubMed\]](#)
- Grefkes, C.; Fink, G.R. Connectivity-based approaches in stroke and recovery of function. *Lancet Neurol.* **2014**, *13*, 206–216. [\[CrossRef\]](#)
- Hachinski, V.; Donnan, G.A.; Gorelick, P.B.; Hacke, W.; Cramer, S.C.; Kaste, M.; Fisher, M.; Brainin, M.; Buchan, A.M.; Lo, E.H.; et al. Stroke: Working toward a prioritized world agenda. *Stroke* **2010**, *41*, 1084–1099. [\[CrossRef\]](#) [\[PubMed\]](#)
- Gao, W.; Cui, Z.; Yu, Y.; Mao, J.; Xu, J.; Ji, L.; Kan, X.; Shen, X.; Li, X.; Zhu, S.; et al. Application of a Brain–Computer Interface System with Visual and Motor Feedback in Limb and Brain Functional Rehabilitation after Stroke: Case Report. *Brain Sci.* **2022**, *12*, 1083. [\[CrossRef\]](#)
- Jin, J.; Qu, T.; Xu, R.; Wang, X.; Cichocki, A. Motor Imagery EEG Classification Based on Riemannian Sparse Optimization and Dempster-Shafer Fusion of Multi-Time-Frequency Patterns. *IEEE Trans. Neural Syst. Rehabil. Eng.* **2022**, *31*, 58–67. [\[CrossRef\]](#)
- Al-Qazzaz, N.K.; Alyasseri, Z.A.A.; Abdulkareem, K.H.; Ali, N.S.; Al-Mhiqani, M.N.; Guger, C. EEG feature fusion for motor imagery: A new robust framework towards stroke patients rehabilitation. *Comput. Biol. Med.* **2021**, *137*, 104799. [\[CrossRef\]](#)
- Al-Timemy, A.H.; Bugmann, G.; Outram, N.; Escudero, J. Reduction in classification errors for myoelectric control of hand movements with independent component analysis. In Proceedings of the The 5th International Conference on Information Technology, ICIT, Bali, Indonesia, 26–28 October 2011; Volume 2011.
- Al-Qazzaz, N.K.; Ali, S.H.B.M.; Ahmad, S.A.; Escudero, J. Optimal EEG Channel Selection for Vascular Dementia Identification Using Improved Binary Gravitation Search Algorithm. In Proceedings of the International Conference for Innovation in Biomedical Engineering and Life Sciences, Penang, Malaysia, 10–13 December 2017; pp. 125–130.
- Al-Qazzaz, N.K.; Sabir, M.K.; Ali, S.H.B.M.; Ahmad, S.A.; Grammer, K. Multichannel optimization with hybrid spectral-entropy markers for gender identification enhancement of emotional-based EEGs. *IEEE Access* **2021**, *9*, 107059–107078. [\[CrossRef\]](#)
- Velasco, I.; Sipols, A.; De Blas, C.S.; Pastor, L.; Bayona, S. Motor imagery EEG signal classification with a multivariate time series approach. *Biomed. Eng. OnLine* **2023**, *22*, 29. [\[CrossRef\]](#)

13. Jouzizadeh, M. EEG-Assessed Network and Signal Variability Features in Males and Females during a Visuospatial Task. Ph.D. Thesis, Université d'Ottawa/University of Ottawa, Ottawa, ON, Canada, 2024.
14. Al-Qazzaz, N.K.; Sabir, M.K.; Ali, S.; Ahmad, S.A.; Grammer, K. Effective EEG channels for emotion identification over the brain regions using differential evolution algorithm. In Proceedings of the 2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Berlin, Germany, 23–27 July 2019; pp. 4703–4706.
15. Dornhege, G.; Millán, J.d.R.; Hinterberger, T.; McFarland, D.J.; Muller, K.R. *Toward Brain-Computer Interfacing*; MIT Press: Cambridge, MA, USA, 2007; Volume 63.
16. Kingphai, K.; Moshfeghi, Y. On channel selection for EEG-based mental workload classification. In Proceedings of the International Conference on Machine Learning, Optimization, and Data Science, Grasmere, UK, 22–26 September 2023; pp. 403–417.
17. Liu, T.; Wu, Y.; Ye, A.; Cao, L.; Cao, Y. Two-stage sparse multi-objective evolutionary algorithm for channel selection optimization in BCIs. *Front. Hum. Neurosci.* **2024**, *18*, 1400077. [[CrossRef](#)] [[PubMed](#)]
18. Soleimani, B.; Dallasta, I.; Das, P.; Kulasingham, J.P.; Girgenti, S.; Simon, J.Z.; Babadi, B.; Marsh, E.B. Altered directional functional connectivity underlies post-stroke cognitive recovery. *Brain Commun.* **2023**, *5*, fcad149. [[CrossRef](#)] [[PubMed](#)]
19. Seguin, C.; Sporns, O.; Zalesky, A. Brain network communication: Concepts, models and applications. *Nat. Rev. Neurosci.* **2023**, *24*, 557–574. [[CrossRef](#)] [[PubMed](#)]
20. Mang, J.; Xu, Z.; Qi, Y.; Zhang, T. Favoring the cognitive-motor process in the closed-loop of BCI mediated post stroke motor function recovery: Challenges and approaches. *Front. Neurorobotics* **2023**, *17*, 1271967. [[CrossRef](#)] [[PubMed](#)]
21. Nunez, P.L.; Srinivasan, R.; Westdorp, A.F.; Wijesinghe, R.S.; Tucker, D.M.; Silberstein, R.B.; Cadusch, P.J. EEG coherency: I: Statistics, reference electrode, volume conduction, Laplacians, cortical imaging, and interpretation at multiple scales. *Electroencephalogr. Clin. Neurophysiol.* **1997**, *103*, 499–515. [[CrossRef](#)] [[PubMed](#)]
22. Al-Qazzaz, N.K.; Hamid Bin Mohd Ali, S.; Ahmad, S.A.; Islam, M.S.; Escudero, J. Automatic artifact removal in EEG of normal and demented individuals using ICA–WT during working memory tasks. *Sensors* **2017**, *17*, 1326. [[CrossRef](#)] [[PubMed](#)]
23. Al-Qazzaz, N.K.; Ali, S.H.B.M.; Ahmad, S.A.; Islam, M.S.; Escudero, J. Discrimination of stroke-related mild cognitive impairment and vascular dementia using EEG signal analysis. *Med. Biol. Eng. Comput.* **2018**, *56*, 137–157. [[CrossRef](#)] [[PubMed](#)]
24. Zhang, Y.; Nam, C.S.; Zhou, G.; Jin, J.; Wang, X.; Cichocki, A. Temporally constrained sparse group spatial patterns for motor imagery BCI. *IEEE Trans. Cybern.* **2018**, *49*, 3322–3332. [[CrossRef](#)] [[PubMed](#)]
25. Chaudhary, S.; Taran, S.; Bajaj, V.; Siuly, S. A flexible analytic wavelet transform based approach for motor-imagery tasks classification in BCI applications. *Comput. Methods Programs Biomed.* **2020**, *187*, 105325. [[CrossRef](#)]
26. Duan, L.; Hongxin, Z.; Khan, M.S.; Fang, M. Recognition of motor imagery tasks for BCI using CSP and chaotic PSO twin SVM. *J. China Univ. Posts Telecommun.* **2017**, *24*, 83–90. [[CrossRef](#)]
27. Alotaiby, T.; Abd El-Samie, F.E.; Alshebeili, S.A.; Ahmad, I. A review of channel selection algorithms for EEG signal processing. *EURASIP J. Adv. Signal Process.* **2015**, *2015*, 66. [[CrossRef](#)]
28. He, L.; Yu, Z.; Gu, Z.; Li, Y. Bhattacharyya bound based channel selection for classification of motor imageries in EEG signals. In Proceedings of the 2009 Chinese Control and Decision Conference, Guilin, China, 17–19 June 2009; pp. 2353–2356.
29. Tam, W.K.; Ke, Z.; Tong, K.Y. Performance of common spatial pattern under a smaller set of EEG electrodes in brain-computer interface on chronic stroke patients: A multi-session dataset study. In Proceedings of the 2011 annual International Conference of the IEEE Engineering in Medicine and Biology Society, Boston, MA, USA, 30 August–3 September 2011; pp. 6344–6347.
30. Gómez-López, J.C.; Escobar, J.J.; Díaz, A.F.; Damas, M.; Gil-Montoya, F.; González, J. Boosting the convergence of a GA-based wrapper for feature selection problems on high-dimensional data. In Proceedings of the Genetic and Evolutionary Computation Conference Companion, Boston, MA, USA, 9–13 July 2022; pp. 431–434.
31. Shen, C.; Zhang, K. Two-stage improved Grey Wolf optimization algorithm for feature selection on high-dimensional classification. *Complex Intell. Syst.* **2022**, *8*, 2769–2789. [[CrossRef](#)]
32. Tiwari, A.; Chaturvedi, A. Automatic EEG channel selection for multiclass brain-computer interface classification using multiobjective improved firefly algorithm. *Multimed. Tools Appl.* **2023**, *82*, 5405–5433. [[CrossRef](#)]
33. Millán, J.d.R.; Franzé, M.; Mouriño, J.; Cincotti, F.; Babiloni, F. Relevant EEG features for the classification of spontaneous motor-related tasks. *Biol. Cybern.* **2002**, *86*, 89–95. [[CrossRef](#)] [[PubMed](#)]
34. Martínez-Cagigal, V.; Santamaría-Vázquez, E.; Hornero, R. Brain–computer interface channel selection optimization using meta-heuristics and evolutionary algorithms. *Appl. Soft Comput.* **2022**, *115*, 108176. [[CrossRef](#)]
35. Gaur, P.; Gupta, H.; Chowdhury, A.; McCreadie, K.; Pachori, R.B.; Wang, H. A sliding window common spatial pattern for enhancing motor imagery classification in EEG-BCI. *IEEE Trans. Instrum. Meas.* **2021**, *70*, 4002709. [[CrossRef](#)]
36. Shi, B.; Wang, Q.; Yin, S.; Yue, Z.; Huai, Y.; Wang, J. A binary harmony search algorithm as channel selection method for motor imagery-based BCI. *Neurocomputing* **2021**, *443*, 12–25. [[CrossRef](#)]
37. Ghaemi, A.; Rashedi, E.; Pourrahimi, A.M.; Kamandar, M.; Rahdari, F. Automatic channel selection in EEG signals for classification of left or right hand movement in Brain Computer Interfaces using improved binary gravitation search algorithm. *Biomed. Signal Process. Control* **2017**, *33*, 109–118. [[CrossRef](#)]
38. Lotte, F.; Congedo, M.; Lécuyer, A.; Lamarche, F.; Arnaldi, B. A review of classification algorithms for EEG-based brain–computer interfaces. *J. Neural Eng.* **2007**, *4*, R1. [[CrossRef](#)]
39. Hassan, M.M.; Alam, M.G.R.; Uddin, M.Z.; Huda, S.; Almogren, A.; Fortino, G. Human emotion recognition using deep belief network architecture. *Inf. Fusion* **2019**, *51*, 10–18. [[CrossRef](#)]

40. Khosla, A.; Khandnor, P.; Chand, T. A comparative analysis of signal processing and classification methods for different applications based on EEG signals. *Biocybern. Biomed. Eng.* **2020**, *40*, 649–690. [\[CrossRef\]](#)
41. Altan, G.; Kutlu, Y.; Allahverdi, N. Deep belief networks based brain activity classification using EEG from slow cortical potentials in stroke. *Int. J. Appl. Math. Electron. Comput.* **2016**, *4*, 205–210. [\[CrossRef\]](#)
42. García-Laencina, P.J.; Rodríguez-Bermudez, G.; Roca-Dorda, J. Exploring dimensionality reduction of EEG features in motor imagery task classification. *Expert Syst. Appl.* **2014**, *41*, 5285–5295. [\[CrossRef\]](#)
43. Aler, R.; Galván, I.M. Optimizing the number of electrodes and spatial filters for Brain–Computer Interfaces by means of an evolutionary multi-objective approach. *Expert Syst. Appl.* **2015**, *42*, 6215–6223. [\[CrossRef\]](#)
44. Lal, T.N.; Schroder, M.; Hinterberger, T.; Weston, J.; Bogdan, M.; Birbaumer, N.; Scholkopf, B. Support vector channel selection in BCI. *IEEE Trans. Biomed. Eng.* **2004**, *51*, 1003–1010. [\[CrossRef\]](#) [\[PubMed\]](#)
45. Rakotomamonjy, A.; Guigue, V. BCI competition III: Dataset II-ensemble of SVMs for BCI P300 speller. *IEEE Trans. Biomed. Eng.* **2008**, *55*, 1147–1154. [\[CrossRef\]](#) [\[PubMed\]](#)
46. Tan, P.; Wang, X.; Wang, Y. Dimensionality reduction in evolutionary algorithms-based feature selection for motor imagery brain-computer interface. *Swarm Evol. Comput.* **2020**, *52*, 100597. [\[CrossRef\]](#)
47. Tsai, C.F.; Eberle, W.; Chu, C.Y. Genetic algorithms in feature and instance selection. *Knowl.-Based Syst.* **2013**, *39*, 240–247. [\[CrossRef\]](#)
48. Udhaya Kumar, S.; Hannah Inbarani, H. PSO-based feature selection and neighborhood rough set-based classification for BCI multiclass motor imagery task. *Neural Comput. Appl.* **2017**, *28*, 3239–3258. [\[CrossRef\]](#)
49. Baig, M.Z.; Aslam, N.; Shum, H.P.; Zhang, L. Differential evolution algorithm as a tool for optimal feature subset selection in motor imagery EEG. *Expert Syst. Appl.* **2017**, *90*, 184–195. [\[CrossRef\]](#)
50. Atyabi, A.; Luerssen, M.; Fitzgibbon, S.; Powers, D.M. Evolutionary feature selection and electrode reduction for EEG classification. In Proceedings of the 2012 IEEE Congress on Evolutionary Computation, Brisbane, QLD, Australia, 10–15 June 2012; pp. 1–8.
51. Nakisa, B.; Rastgoo, M.N.; Tjondronegoro, D.; Chandran, V. Evolutionary computation algorithms for feature selection of EEG-based emotion recognition using mobile sensors. *Expert Syst. Appl.* **2018**, *93*, 143–155. [\[CrossRef\]](#)
52. Al-Qazzaz, N.K.; Ali, S.H.M.; Ahmad, S.A. Differential Evolution Based Channel Selection Algorithm on EEG Signal for Early Detection of Vascular Dementia among Stroke Survivors. In Proceedings of the 2018 IEEE-EMBS Conference on Biomedical Engineering and Sciences (IECBES), Sarawak, Malaysia, 3–6 December 2018; pp. 239–244.
53. Alyasseri, Z.A.A.; Khader, A.T.; Al-Betar, M.A.; Joao, P.P.; Osama, A.A. EEG-based Person Authentication Using Multi-objective Flower Pollination Algorithm. In Proceedings of the Evolutionary Computation (CEC), Rio de Janeiro, Brazil, 8–13 July 2018; pp. 15–22.
54. Yang, X.S. A new metaheuristic bat-inspired algorithm. In *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 65–74.
55. Hyvarinen, A. Fast and robust fixed-point algorithms for independent component analysis. *IEEE Trans. Neural Netw.* **1999**, *10*, 626–634. [\[CrossRef\]](#) [\[PubMed\]](#)
56. Escudero, J.; Hornero, R.; Abásolo, D.; Fernández, A. Blind source separation to enhance spectral and non-linear features of magnetoencephalogram recordings. Application to Alzheimer’s disease. *Med. Eng. Phys.* **2009**, *31*, 872–879. [\[CrossRef\]](#) [\[PubMed\]](#)
57. James, C.J.; Hesse, C.W. Independent component analysis for biomedical signals. *Physiol. Meas.* **2005**, *26*, R15. [\[CrossRef\]](#) [\[PubMed\]](#)
58. Al-Qazzaz, N.K.; Hamid Bin Mohd Ali, S.; Ahmad, S.A.; Islam, M.S.; Escudero, J. Selection of mother wavelet functions for multi-channel EEG signal analysis during a working memory task. *Sensors* **2015**, *15*, 29015–29035. [\[CrossRef\]](#) [\[PubMed\]](#)
59. Escudero, J.; Hornero, R.; Abásolo, D.; Fernández, A. Quantitative evaluation of artifact removal in real magnetoencephalogram signals with blind source separation. *Ann. Biomed. Eng.* **2011**, *39*, 2274–2286. [\[CrossRef\]](#) [\[PubMed\]](#)
60. Barbati, G.; Porcaro, C.; Zappasodi, F.; Rossini, P.M.; Tecchio, F. Optimization of an independent component analysis approach for artifact identification and removal in magnetoencephalographic signals. *Clin. Neurophysiol.* **2004**, *115*, 1220–1232. [\[CrossRef\]](#)
61. Mammone, N.; La Foresta, F.; Morabito, F.C. Automatic artifact rejection from multichannel scalp EEG by wavelet ICA. *IEEE Sens. J.* **2011**, *12*, 533–542. [\[CrossRef\]](#)

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.

Article

Classification and Regression of Pinhole Corrosions on Pipelines Based on Magnetic Flux Leakage Signals Using Convolutional Neural Networks

Yufei Shen  and Wenxing Zhou * 

Department of Civil & Environmental Engineering, The University of Western Ontario,
London, ON N6A 5B9, Canada

* Correspondence: wzhou@eng.uwo.ca

Abstract: Pinhole corruptions on oil and gas pipelines are difficult to detect and size and, therefore, pose a significant challenge to the pipeline integrity management practice. This study develops two convolutional neural network (CNN) models to identify pinholes and predict the sizes and location of the pinhole corruptions according to the magnetic flux leakage signals generated using the magneto-static finite element analysis. Extensive three-dimensional parametric finite element analysis cases are generated to train and validate the two CNN models. Additionally, comprehensive algorithm analysis evaluates the model performance, providing insights into the practical application of CNN models in pipeline integrity management. The proposed classification CNN model is shown to be highly accurate in classifying pinholes and pinhole-in-general corrosion defects. The proposed regression CNN model is shown to be highly accurate in predicting the location of the pinhole and obtain a reasonably high accuracy in estimating the depth and diameter of the pinhole, even in the presence of measurement noises. This study indicates the effectiveness of employing deep learning algorithms to enhance the integrity management practice of corroded pipelines.

Keywords: deep learning algorithm; convolutional neural network; pipeline; corrosion; pinhole; finite element analysis; magnetic flux leakage signal



Citation: Shen, Y.; Zhou, W. Classification and Regression of Pinhole Corrosions on Pipelines Based on Magnetic Flux Leakage Signals Using Convolutional Neural Networks. *Algorithms* **2024**, *17*, 347. <https://doi.org/10.3390/a17080347>

Academic Editors: Chih-Lung Lin, Bor-Jiunn Hwang, Shaou-Gang Miaou and Chi-Hung Chuang

Received: 24 June 2024
Revised: 1 August 2024
Accepted: 6 August 2024
Published: 8 August 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Pipelines are economical and safe means for transporting and distributing large volumes of oil and gas products across great distances [1]. However, pipeline failures can lead to severe consequences such as fatalities, economic losses, and environmental damages [2]. Among the various failure mechanisms threatening the structural integrity of pipelines, corrosion is a leading threat [3]. The in-line inspection (ILI) tool is widely used for detecting and measuring corruptions to assess pipeline integrity [4], and magnetic flux leakage (MFL) is the predominant ILI technology for identifying internal and external corrosion defects in both liquid and gas pipelines [5]. The Pipeline Operators Forum (POF) suggests that a single corrosion anomaly on the internal or external surface of a pipeline can be categorized into one of seven classes based on the ILI-reported corrosion length and width as illustrated in Figure 1 [6], i.e., axial slotting, circumferential slotting, axial grooving, circumferential grooving, general, pinhole and pitting, to better quantify the accuracy of ILI tools and facilitate the selection of appropriate models to evaluate the burst capacity of pipelines containing corrosion defects [6,7].

The estimation of the corrosion anomaly sizes based on MFL signals falls into the inverse modeling realm. The pipeline industry frequently employs the iterative inverse model because of its remarkable precision [8–10]. However, the iterative inverse model is computationally costly. Inverse models based on machine learning provide promising alternatives to sizing corrosion defects from MFL signals with high accuracy but are more computationally efficient compared to the iterative inverse models [11–13]. For instance,

Kandroodi et al. [14] fed the algorithm-estimated defect width and length, as well as the extracted peak-to-peak values from the MFL signals, into a Gaussian radial basis function neural network to predict the corrosion depth. Feng et al. [15] proposed an error adjustment methodology for reconstructing the corrosion profiles from MFL signals based on a radial basis function neural network. However, these models have certain limitations: the training of these models relies on manually chosen features from the MFL signals, potentially omitting valuable data necessary for accurately predicting the corrosion profile. Additionally, the majority of these techniques focus solely on one MFL signal dimension, neglecting the critical information present in the other two dimensions.

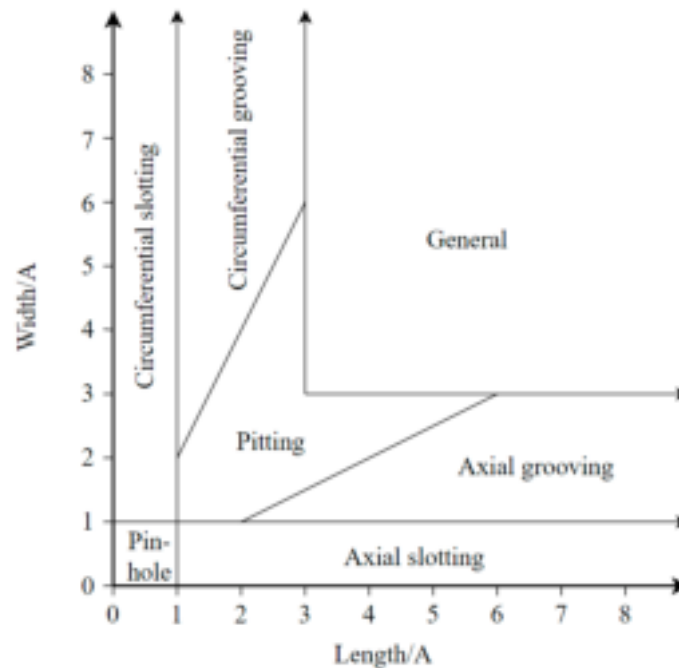


Figure 1. Seven corrosion anomaly categories are based on the anomaly length and width. Note: $A = \max\{10 \text{ mm}, \text{pipe wall thickness}\}$.

Over the past decade, convolutional neural networks (CNNs) have risen as a revolutionary deep learning algorithm, yielding remarkable results across various pattern recognition fields, including image and voice processing [16,17]. The CNN tool can automatically extract deep features from input data in place of manually designed feature extractors; therefore, it is well suited to solve the inverse modeling based on the MFL signals. The application of CNN in the inverse analysis of MFL signals has been reported in the literature. Lu et al. [18] introduced the visual transformation CNN, which predicts the length, width, and depth of anomalies by taking into account two components (i.e., circumferential and radial) of the MFL signal while disregarding the longitudinal component. Shen and Zhou [19] employed CNN to estimate the locations and dimensions of corruptions on steel pipelines based on three components of the MFL signals. Wang and Chen [20] proposed a CNN architecture comprising two main modules: one for anomaly classification and another for anomaly size regression. The input of the classification module consists of all three components of the MFL signal, and the outputs are seven anomaly categories, as illustrated in Figure 1. The regression module incorporates seven separate CNNs, each tailored to a specific anomaly type, predicting dimensions such as width, length, and depth of the defect.

According to a report from the European Gas Pipeline Incident Data Group (EGIG) [21], pinhole corrosion is identified as a major contributor to pipeline failures due to its potential to evolve from a small initial volume of metal loss into more severe metal loss, given the challenge in detection via pressure monitoring, particularly when the leaked products do not exceed ground level [22,23]. Despite the impact of pinholes on pipeline integrity, studies

that are specifically focused on detecting and sizing pinhole corruptions are limited in the literature, thus creating a knowledge gap. Furthermore, small corrosion anomalies such as pinholes may sometimes overlap with a relatively large corrosion area [23–25], creating a complex defect known in this study as a pinhole-in-general corrosion (PIC) defect. There are also few studies in the literature investigating the detection and sizing of PIC defects.

The objective of this study is to develop deep learning algorithms, i.e., CNN models, to classify and size pinholes and PIC defects based on a large set of MFL signals. The classification CNN model deals with six scenarios, i.e., pinholes on the internal or external pipe surface, general corruptions on the internal or external surface, and PIC defects on the internal or external surface. The signals for the pinholes and PIC defects identified by the classification model are then fed into the regression CNN model to predict the sizes and location of the pinhole and PIC defects. In practice, it is not necessary to apply the two CNN models sequentially. Depending on the specific application, data can be fed directly into the regression model if the classification has already been carried out through other means and the primary goal is to predict the size of the defect. Since real MFL signals are typically proprietary and not publicly accessible, the MFL signals used in this study are numerically generated by applying the three-dimensional (3D) magneto-static finite element analyses (FEA) using the commercial software COMSOL (Version 5.6). The three-dimensional (i.e., longitudinal, circumferential, and radial) MFL signal maps serve as inputs for training and validating CNN models to classify the defects and predict the sizes, as well as the circumferential and longitudinal positions, of pinholes on the pipeline. The impact of the measurement noises associated with the MFL signal on the accuracy of the regression CNN model is also investigated. This research fills a knowledge gap in the literature by applying CNN to detect, locate, and size pinholes and pinhole-in-corrosion defects. It is noted that accurately identifying the position of the pinhole within the general corrosion is crucial as the relative position of the pinhole within the general corrosion has a marked effect on the burst capacity of the PIC defect and, therefore, has strong implications for the accurate prediction of the burst capacity. This study demonstrates the effectiveness of CNN for the detection and sizing of pinhole corrosion and the viability of CNN for applications in pipeline integrity management practice.

The rest of the paper is structured as follows. Section 2 introduces the principles of the MFL technology and essential background information on the relationship between MFL signals and anomaly features. Section 3 introduces the proposed FEA model and the simulation parameters of MFL signals. Section 4 describes the structure of the developed CNN models and their application to the classification and regression of pinholes. The corresponding predictive accuracy of the CNN models is also discussed in Section 4. Section 5 summarizes the concluding remarks of the study.

2. Principles of MFL Technique

The magneto-statics FEA is employed to numerically generate the MFL signals in the study. The corresponding governing equation derived from Maxwell's equations is shown in Equation (1):

$$\nabla \times \left(\frac{1}{\mu} \nabla \times \mathbf{A} \right) = 0 \quad (1)$$

where $\nabla \times$ denotes the curl operation; μ represents the magnetic permeability of the ferromagnetic specimen (N/A^2), and $\mathbf{A}$ is the magnetic potential vector. Equation (1) is solved under the specified boundary conditions to obtain $\mathbf{A}$, which is then used to obtain the magnetic flux density $\mathbf{B}$ as follows [26,27]:

$$\mathbf{B} = \nabla \times \mathbf{A} \quad (2)$$

The MFL technique relies on the measurement principle where a ferromagnetic material demonstrates MFL in the vicinity of a defect, as illustrated in Figure 2 [18]. The measurement procedure is as follows: during the ILI process, the MFL tool is propelled

through the pipeline, and magnets within the tool generate a magnetic field. In a perfect pipe, all magnetic flux remains within the pipe wall [28], resulting in a uniform magnetic field distribution [29]. However, in the presence of defects, some magnetic flux “leaks” outside the pipe, forming a magnetic leakage field close to the defect [30]. This leakage field is detected by Hall sensors, generating electrical signals [29]. The strength and distribution of the flux leakage vary depending on the geometry of the defect on the pipeline. By capturing the MFL signal, which typically consists of a three-dimensional magnetic flux density (B) vector field, represented by B_x in the circumferential direction of the pipe, B_y in the radial direction, and B_z in the longitudinal (axial) direction, it is possible to estimate the size and location characteristics of the corrosion defect.

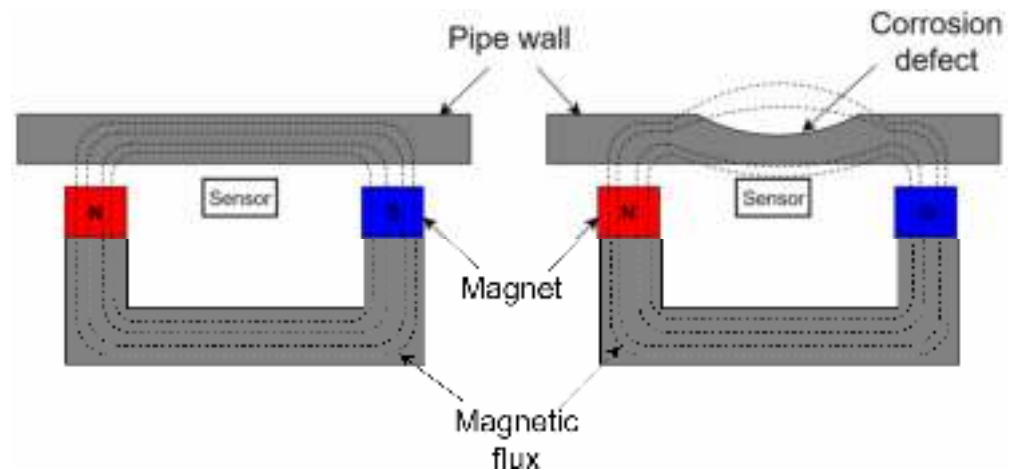


Figure 2. Principle of MFL technique.

Axial (AMFL) and circumferential (CMFL) MFL tools are the two types of MFL tools categorized by the arrangement of the magnets inside the MFL tool. In AMFL tools, the permanent magnets are oriented parallel to the longitudinal axis of the pipe. These tools excel in detecting and measuring defects in the circumferential direction of the pipe but are less effective in sizing features oriented axially [31]. Conversely, CMFL tools have magnets positioned in the circumferential direction of the pipeline. They are more precise in measuring features aligned along the longitudinal direction compared to AMFL tools. The CMFL tool is the main focus of this study because it is more accurate than the AMFL tool for sizing longitudinally-oriented corrosion anomalies, which are perpendicular to the hoop stress resulting from the pipe’s internal pressure and, therefore, can have a great influence on the pipeline burst capacity.

3. Simulating MFL Signals Using FEA

3.1. Simulation Parameters

The CMFL tool considered in this study includes three pairs of permanent magnets. NdFeB is selected as the magnetic material with a high coercive force of 895,000 A/m, stable magnetic properties, and high magnetic energy [32]. The steel used to simulate the pipe is assumed to be grade X52, characterized by a remanent flux density (B_r) of 1T, an electrical conductivity of 5.882×10^6 S/m, coercive field strength (H_c) of 415 A/m, and saturation flux density (B_s) of 1.9T. Figure 3 shows the schematics of a cross-section of the CMFL tool and pipe section, and the geometric and material properties of each component are summarized in Table 1. It is noted that the lift-off value is assumed to be 5 mm, which is the distance between the sensor and the pipe’s internal surface [33].

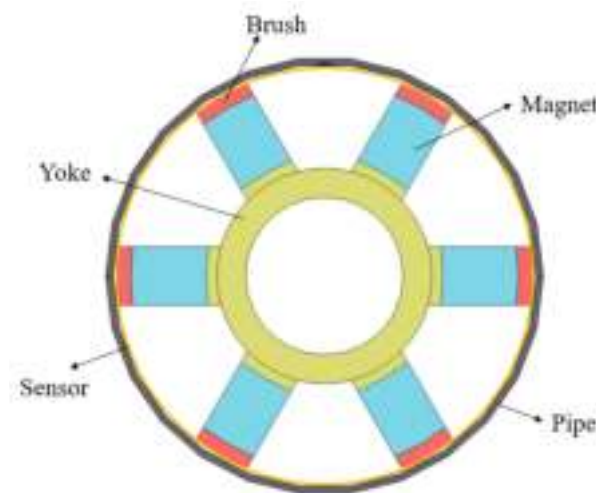


Figure 3. Illustration of CMFL tool.

Table 1. Geometry of every component within the proposed 3D CMFL model.

Elements	Pipe Section	Magnets	Brushes	Yoke
geometry (mm)	$600 \times 610 \times 10^a$	$300 \times 81 \times 105^b$	$300 \times 81 \times 20^b$	$300 \times 300 \times 40^a$

^a Length $\times$ outside diameter $\times$ wall thickness. ^b Length $\times$ width $\times$ thickness.

3.2. FEA Cases

A series of FEA models are created and analyzed with the same magnetization conditions and pipe attributes, but they vary in corrosion types, geometric parameters, and locations in this study. The simulation involves 9600 analysis cases, with 1600 cases allocated to each of the six scenarios as illustrated in Figure 4, namely internal general corrosion (G_{in}), external general corrosion (G_{ex}), internal pinhole (P_{in}), external pinhole (P_{ex}), internal PIC defect (PIC_{in}), and external PIC defect (PIC_{ex}). The general corrosion is idealized as cuboidal-shaped, defined by the width (w_g), depth (d_g), and length (l_g), and the pinhole is idealized as cylindrical-shaped, defined by the radius (r_p) and depth (d_p). Note the center of the pinhole in a PIC defect is assumed to be coincident with that of the general corrosion, which is defined by two coordinates, i.e., its longitudinal location (h) and circumferential location (ϕ), within a 160×160 mm designated area as shown in Figure 5. This area is marked in light blue, with its center specified by the coordinates ($r, \phi = 0, h = 0$), where r represents the inner radius of the pipe. This specific 160×160 mm square area on the inner pipe wall was selected because it encompasses the largest area, showing a consistent magnetic flux density profile between the neighboring magnets.

The analysis cases involving general corrosion only, i.e., G_{in} and G_{ex} , include four values of w_g , i.e., 30, 50, 70, and 90 mm, four values of d_g/t , i.e., 20, 40, 60, and 80%, where t denotes the pipe wall thickness, and four values of l_g , i.e., 30, 50, 70, and 90 mm. This results in 64 general corruptions in total with varying sizes. In terms of the corrosion location, five values of h (−50, −25, 0, 25, and 50 mm) and five values of ϕ (−10, −5, 0, 5, and 10 degrees) are considered. The permutations of 64 defects and 25 locations lead to 1600 cases for each of G_{in} and G_{ex} .

The analysis cases involving pinhole corrosion only, i.e., P_{in} and P_{ex} , include eight values of r_p , i.e., 1, 1.5, 2, 2.5, 3, 3.5, 4, and 4.5 mm, and eight values of d_p/t , i.e., 10, 20, 30, 40, 50, 60, 70, and 80%. This results in 64 pinhole corruptions in total with varying sizes. In terms of the corrosion location, five values of h (−50, −25, 0, 25, and 50 mm) and five values of ϕ (−10, −5, 0, 5, and 10 degrees) are considered. The permutation of 64 defects and 25 locations also leads to 1600 cases for each of P_{in} and P_{ex} .

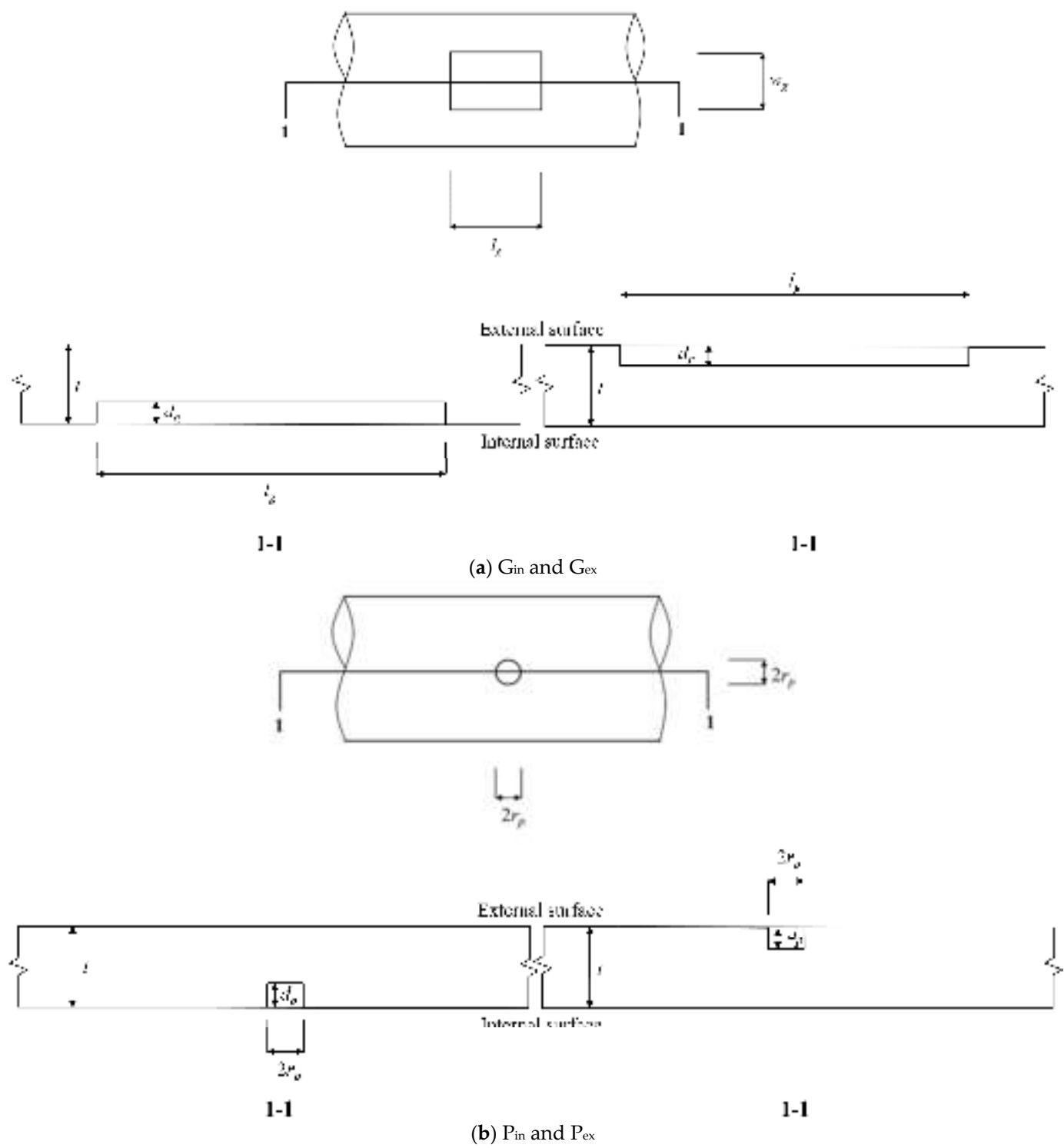


Figure 4. Cont.

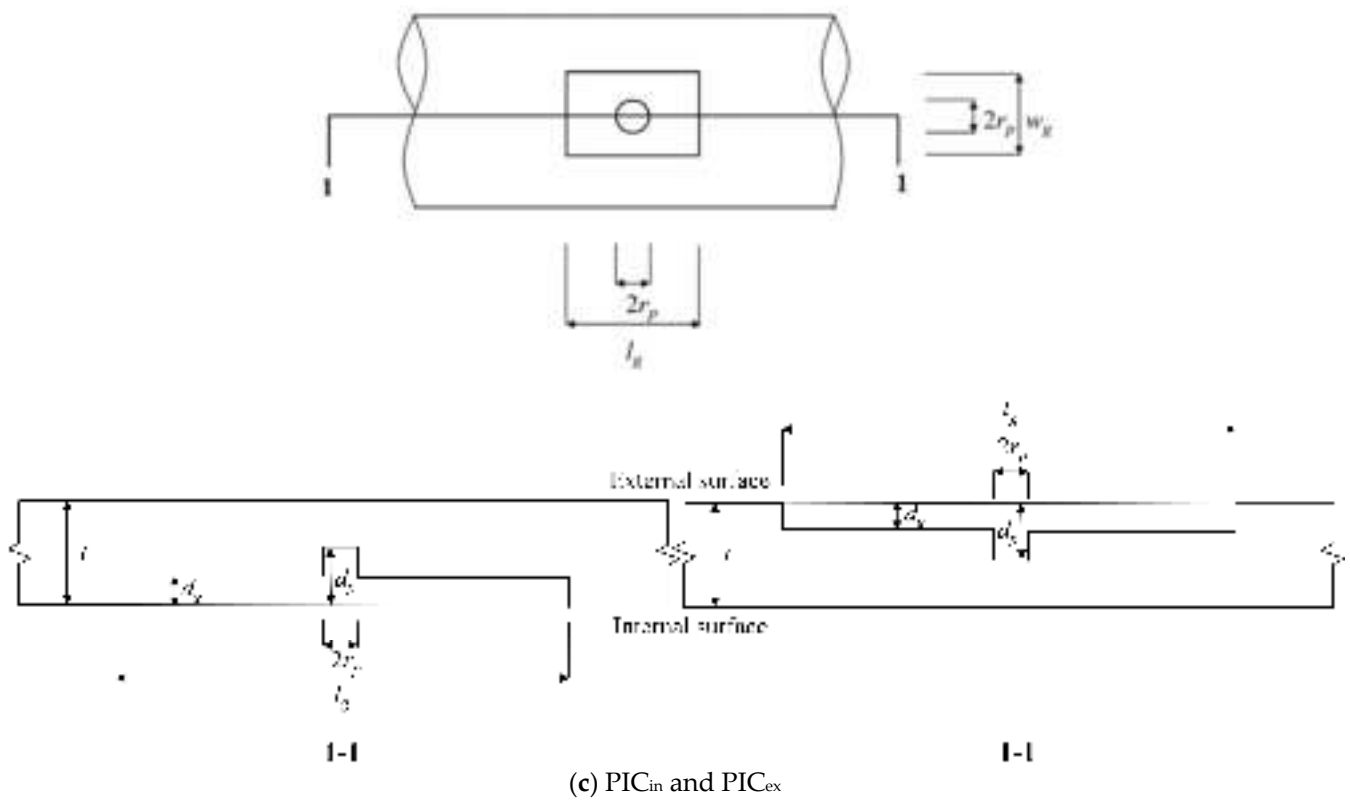


Figure 4. Six types of corrosion situations: (a) G_{in} and G_{ex} ; (b) P_{in} and P_{ex} ; (c) PIC_{in} and PIC_{ex}.

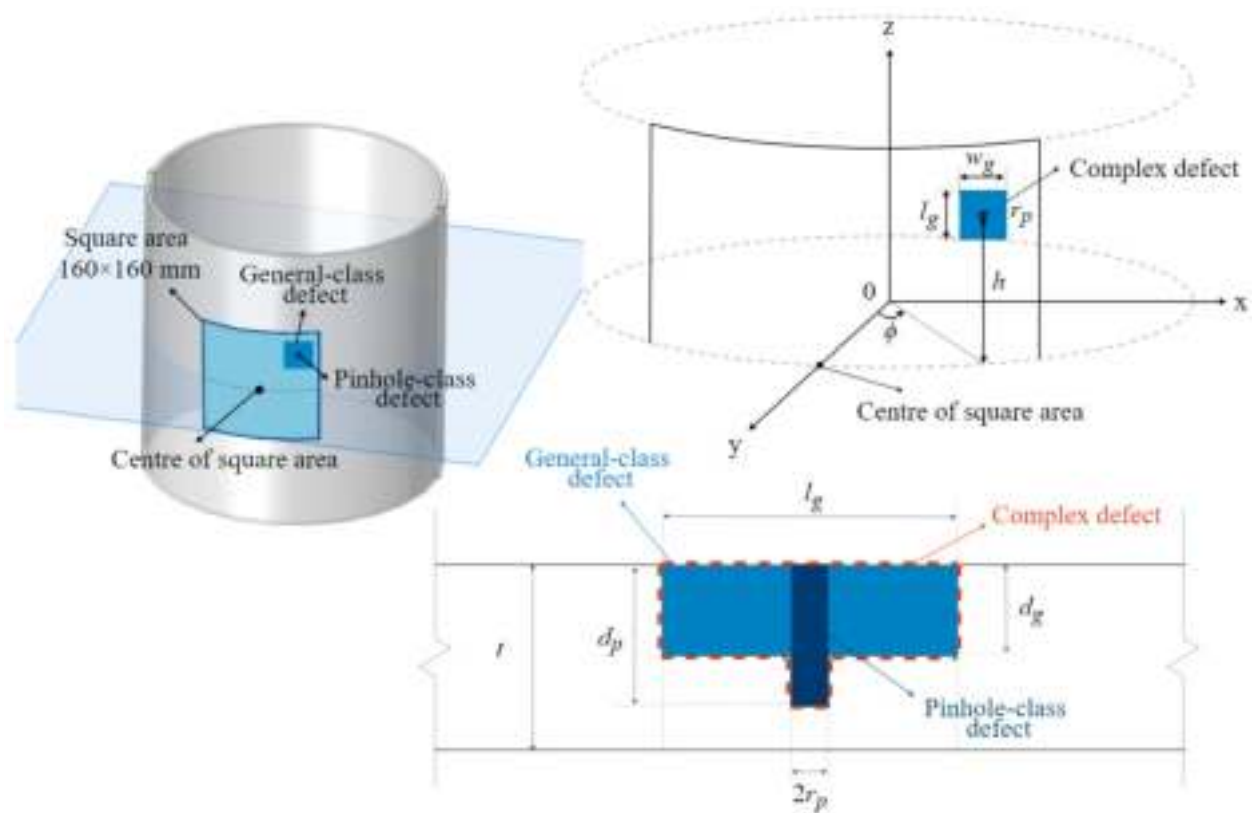


Figure 5. Cylindrical coordinate system defining location parameters h and ϕ .

Analysis cases involving PIC defects, i.e., PIC_{in} and PIC_{ex}, include $w_g = l_g = 50$ mm and four values of r_p , i.e., 1, 2, 3, and 4 mm. In terms of the sizes of general and pinhole corruptions, three groups are considered to capture a wider range of relative relationships between general and pinhole corruptions. The first includes d_g/t equal to 20% and d_p/t equal to 30, 40, 50, 60, 70, or 80%. The second group includes d_g/t equal to 30% and d_p/t equal to 40, 50, 55, 60, 70, or 80%. The last group includes d_g/t equal to 40% and d_p/t equal to 50, 60, 70, or 80%. It should be noted that d_p/t is assumed to be at least 10% larger than d_g/t in all three groups. This results in 64 PIC defects in total with varying sizes. In terms of the corrosion location, five values of h (−50, −25, 0, 25, and 50 mm) and five values of ϕ (−10, −5, 0, 5, and 10 degrees) are considered. The permutation of 64 defects and 25 locations leads to 1600 cases for each of PIC_{in} and PIC_{ex}. It follows that a dataset consisting of 9600 cases ($1600 \times 6 = 9600$) is generated in this study. This large dataset is intended to facilitate the training and validation of the CNN model, enabling comprehensive analysis and robust model development.

The developed FEA model meshes using 4-node tetrahedral elements, where the minimum size is 4 mm, and the maximum mesh size is 55 mm after a convergence study. Specifically, grids inside the 160 mm × 160 mm area are finely resolved at 1 mm intervals. At each grid point, the value of B is calculated via interpolation between two mesh nodes. The system of nonlinear equations is solved using the Newton-Raphson method. As an illustration, Figure 6 depicts the 3D magnetic flux density information (i.e., B_x , B_y , and B_z) within the 160 mm × 160 mm area of a selected FE model. The proposed FEA model was validated by comparing the simulation results with experimental data from a CMFL tool reported by Ireland and Torres [34] and an AMFL tool reported by Azizzadeh and Safizadeh [35]. Readers are referred to Shen and Zhou [19] for details of the validation.

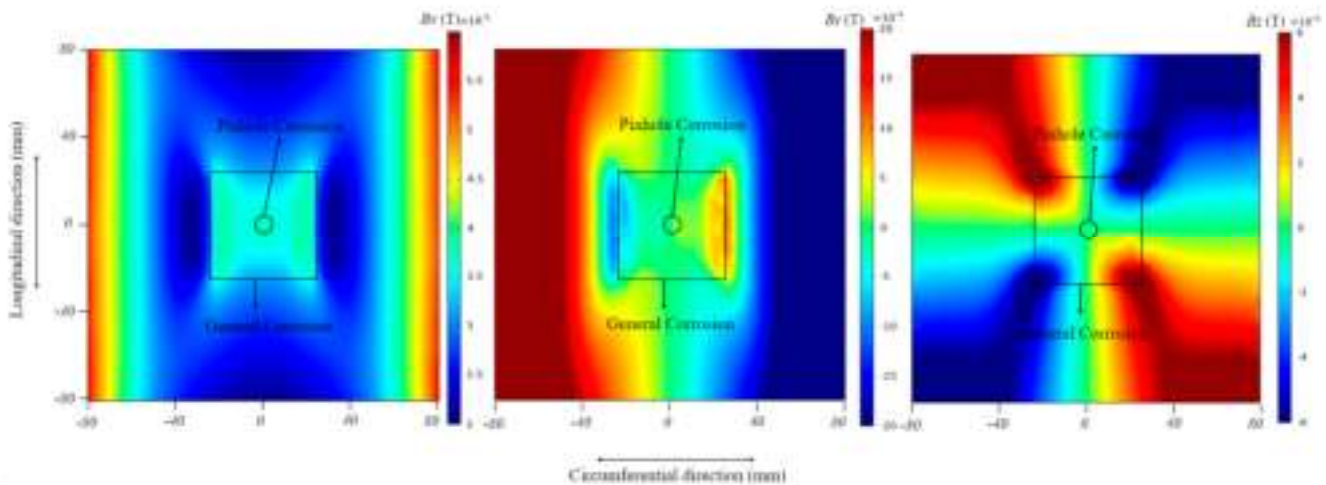


Figure 6. FEA-obtained B_x , B_y , and B_z corresponding to a PIC defect on the internal pipe surface ($w_g = 50$ mm, $l_g = 50$ mm, $d_g/t = 40\%$, $r_p = 4$ mm, $d_p/t = 80\%$, $h = 0$ mm and $\phi = 0$ degree).

MFL signals in practical applications inevitably include noise stemming from factors such as non-uniform pipe wall thickness and differences in sensor lift-off values [36]. These noises can impact the accuracy of the inverse model, necessitating the inclusion of noises in the FEA-simulated MFL signals. The signal-to-noise (SNR) ratio serves as an important parameter for quantifying the noise in the three-dimensional MFL signal. Equation (3) defines SNR in this study [37]:

$$SNR = 10 \log_{10} \left(\frac{\sum_{n=1}^{160} \sum_{m=1}^{160} [B_x(n, m)^2 + B_y(n, m)^2 + B_z(n, m)^2]}{\sum_{n=1}^{160} \sum_{m=1}^{160} [w_x(n, m)^2 + w_y(n, m)^2 + w_z(n, m)^2]} \right) \quad (3)$$

where w_x , w_y and w_z denote the noises along three directions of the pipe, namely circumferential, radial and longitudinal directions, respectively, and (n, m) (where $n, m = 1, 2, \dots, 160$) represents the coordinate of each grid (with 1 mm resolution) in the 160 mm $\times$ 160 mm area, with a total of 25,600 grids. By considering a representative value of SNR equal to 20 [38], three 160 $\times$ 160 matrices of Gaussian-distributed white noises, which are assumed to be independent of each other, are simulated and then applied to B_x , B_y , and B_z , respectively. Each matrix assumes a mean value of zero, with standard deviations adjusted through trial and error until achieving the predefined signal-to-noise ratio. Let σ_n denote the standard deviation of w_x ; the standard deviations of w_y and w_z are assumed to be $0.75\sigma_n$ and $0.25\sigma_n$, respectively. The scaling of σ_n is based on the relative magnitudes of B_x , B_y , and B_z derived from the finite element analysis: for a given analysis case, B_x consistently has the highest magnitude, followed by B_y and B_z , respectively. The final noisy input to the CNN model consists of the three matrices B_x , B_y , and B_z , each combined with respective noisy matrices w_x , w_y , and w_z , resulting in noisy representations of the MFL signals.

It is emphasized that the noisy MFL signal matrices are only fed into the regression CNN model as an extended part in Section 4.2.4. Fan et al. [39] reported that the effect of Gaussian noise on the CNN classification model accuracy is minor. Acharya et al. [40] indicated that removing the noise is not necessary for image classification with deep learning algorithms [41]. Therefore, we only consider the impact of the noises on the accuracy of the regression model in the present study.

4. Convolutional Neural Network

4.1. Classification CNN

4.1.1. Input Information

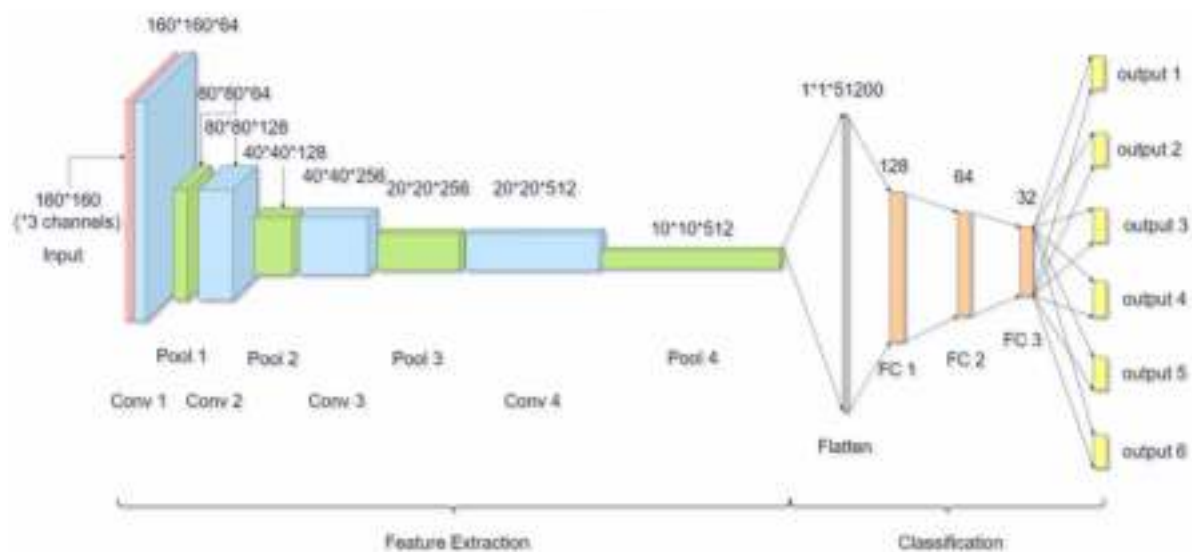
The input to a CNN model is commonly represented as an RGB color image, including three color channels corresponding to green, red, and blue, respectively. This input is compatible with the three-dimensional MFL signals. For a given FE model, the 3D magnetic flux density information (i.e., B_x , B_y , and B_z) within the designated square area is exported from the COMSOL software at the 160 mm $\times$ 160 mm area grids with a resolution of 1 mm. The three generated 160 $\times$ 160 matrices of noise-free MFL signals (one matrix for each dimension) are denoted as the input of the proposed CNN model, and the six types of corrosion (i.e., G_{in} , G_{ex} , P_{in} , P_{ex} , PIC_{in} , and PIC_{ex}) are the output.

4.1.2. Proposed Structure

The proposed classification CNN model in this study comprises a total of 20 layers: layers 1 to 12 are dedicated to feature extraction, while layers 13 to 20 are focused on classification. Layer 20 serves as the output layer, comprising six components that represent the six corrosion types. The detailed information on each layer is summarized in Table 2. As shown in Table 2, the size of the filter in the convolution layer is selected to be (5 $\times$ 5), which is a typical size for large filters [42]. The size of the filter in the maxpooling layer is (2 $\times$ 2), consistent with those commonly used in the existing literature [43,44]. Several dropout layers that randomly drop a certain percentage of neurons from the neural network during the training process are included in the model to avoid overfitting. The employed dropout rate is 0.2, i.e., 20% of the neurons in the neural network are randomly deactivated. Figure 7 depicts the architecture of the proposed classification model with the dimension of each layer, where the input, convolution layers, and max-pooling layers are depicted in pink, blue, and green, respectively. Note that the dropout layers are not shown in Figure 7 as they do not impact the data dimensionality. The thick white line represents the flattened layer, the fully connected layers are represented in orange, and the output layers are shown in yellow in Figure 7.

Table 2. CNN classification model layer information.

Part	Layer No.	Layer Name	Parameters
Feature extraction	1	Convolution	64 filters with size equals to (5×5)
	2	Maxpooling	Pooling size equals to (2×2)
	3	Dropout	Rate = 0.2
	4	Convolution	128 filters with size equals to (5×5)
	5	Maxpooling	Pooling size equals to (2×2)
	6	Dropout	Rate = 0.2
	7	Convolution	256 filters with size equals to (5×5)
	8	Maxpooling	Pooling size equals to (2×2)
	9	Dropout	Rate = 0.2
	10	Convolution	512 filters with size equals to (5×5)
	11	Maxpooling	Pooling size equals to (2×2)
	12	Dropout	Rate = 0.2
Classification	13	Flatten layer	Unit number equals to 51,200
	14	Fully connected layer	Unit number equals to 128
	15	Dropout	Rate = 0.2
	16	Fully connected layer	Unit number equals to 64
	17	Dropout	Rate = 0.2
	18	Fully connected layer	Unit number equals to 32
	19	Dropout	Rate = 0.2
	20	Output 1	Unit number equals to 1
		Output 2	Unit number equals to 1
		Output 3	Unit number equals to 1
		Output 4	Unit number equals to 1
		Output 5	Unit number equals to 1
		Output 6	Unit number equals to 1

**Figure 7.** Proposed CNN classification model structure.

4.1.3. Results

By adopting an 80–20% training-test data ratio, there are 1280 training cases and 320 test cases for each of the six corrosion types. The classification results obtained from the proposed CNN model are summarized in Table 3. Table 3 demonstrates that the proposed model consistently achieves prediction accuracies higher than 0.93 in both the training and test datasets, with the lowest accuracy of 0.934 observed in the training dataset for G_{ex} . It is noted that the model performs better in the test dataset than the training dataset, with accuracy rates of each corrosion type in the test dataset higher than or equal to the rates in the training dataset. This may be due to the regularization techniques, such as dropout, adopted in the CNN model. Although these techniques can prevent overfitting, they may also introduce noise or uncertainty during training, which could temporarily degrade the model's performance on the training data while improving the generalization and performance on the test data [45].

Table 3. Results of the classification CNN model.

Category No.	Category	Total	Train	Test	Misprediction in Train	Misprediction in Test	Train Accuracy	Test Accuracy
0	G_{in}	1600	1280	320	7	0	0.978	1.000
1	G_{ex}	1600	1280	320	21	0	0.934	1.000
2	P_{in}	1600	1280	320	0	0	1.000	1.000
3	P_{ex}	1600	1280	320	2	0	0.994	1.000
4	PIC_{in}	1600	1280	320	1	0	0.997	1.000
5	PIC_{ex}	1600	1280	320	1	1	0.997	0.997

Notably, 32 cases in the training dataset and one case in the test dataset have been misclassified, resulting in a misprediction rate of 0.34% ($33/9600 = 0.34\%$), demonstrating the excellent accuracy of the developed model. Figure 8 illustrates the distribution of mispredictions in the training dataset across different categories. For instance, 65.6% (i.e., 21) of misclassified training cases belong to the G_{ex} category. Among these, four cases are misclassified as P_{in} , and 17 cases are misclassified as P_{ex} . Furthermore, six out of the 33 cases have misclassified locations (internal/external surface of the pipe), i.e., two P_{ex} cases misclassified as P_{in} , and four G_{ex} cases misclassified as P_{in} .

Note that the model achieves better classification accuracy for P_{in} , P_{ex} , PIC_{in} , and PIC_{ex} than for G_{in} and G_{ex} . The values of d_g/t included in the entire dataset for all six types of corruptions are 20, 30, 40, 60, and 80%, while the values of d_p/t included in the entire dataset are 20, 30, 40, 50, 55, 60, 70 and 80%. The wider range of the corrosion depth for cases involving pinholes (i.e., P_{in} , P_{ex} , PIC_{in} , and PIC_{ex}) compared to the cases involving general corrosion (G_{in} and G_{ex}) allows the model to learn a richer set of features and variations associated with pinhole corrosion, making the model more adept at distinguishing between different types and severity of pinhole corrosion. All of the 28 misclassified cases in the G_{in} and G_{ex} categories involve relatively shallow corruptions, among which 26 cases have $d_g/t = 20\%$ and two cases have $d_g/t = 40\%$. The distribution of the 28 misclassified cases by w_g/l_g is shown in Figure 9. Figure 9 indicates that the classification CNN model is less accurate for wide general corruptions with w_g/l_g larger than 1.67. It is also noteworthy that the developed model is less accurate for cases containing external corruptions (i.e., G_{ex} , P_{ex} , and PIC_{ex}) than for cases involving internal corruptions (i.e., G_{in} , P_{in} , and PIC_{in}). A potential explanation is that the model input (i.e., the MFL signal map) for external defects has less pronounced features because the variation in B due to the metal-loss defect on the external pipe surface is smaller than that caused by the internal metal-loss defect as the external defect is farther away from the sensor than the internal defect.

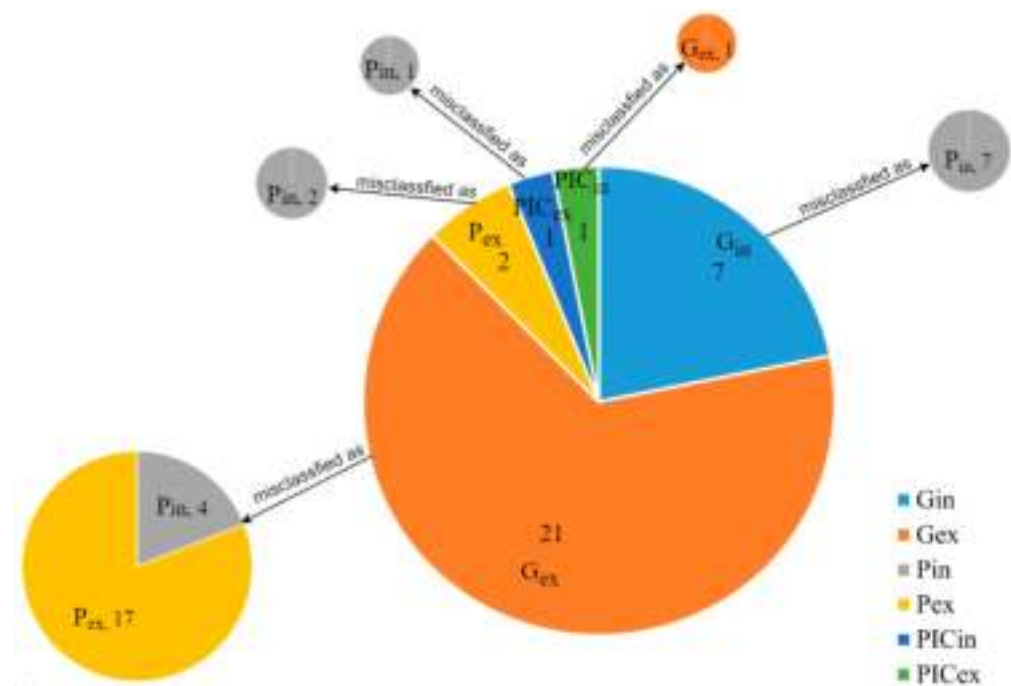


Figure 8. Distribution of misclassifications in the training dataset.

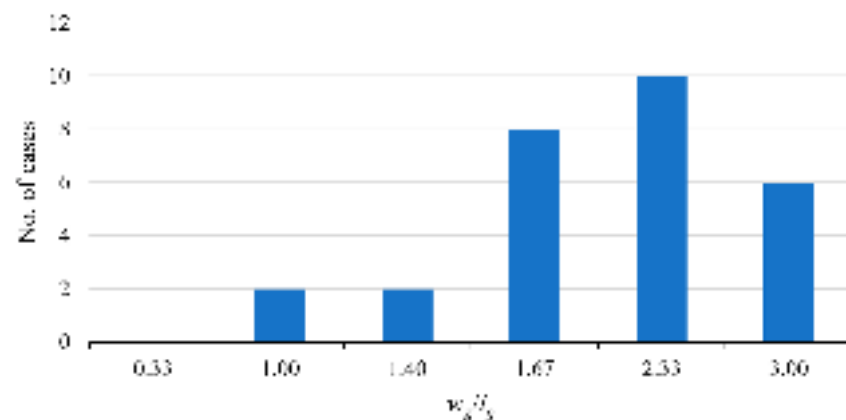


Figure 9. Distribution of misclassified cases in G_{in} and G_{ex} by w_g/l_g .

4.2. Regression CNN

4.2.1. Input Information

The results of the classification CNN model are utilized to extract correctly classified cases for P_{in} , P_{ex} , PIC_{in} , and PIC_{ex} , excluding four misclassifications in the training dataset and one misclassification in the test dataset, resulting in 1600, 1598, 1599, and 1598 input cases for P_{in} , P_{ex} , PIC_{in} , and PIC_{ex} , respectively, to train and validate the regression CNN model. The three generated 160×160 matrices of noise-free MFL signals for each of the 6395 ($=1600 + 1598 + 1599 + 1598$) cases are the input to the developed regression model, and the predicted r_p , d_p/t , h , and ϕ values for each case are the output. The correctly classified data are adopted to train the regression CNN model because including misclassified data (such as a general corrosion anomaly misclassified as a pinhole) in training is problematic, e.g., defining an error function that is applicable to all input data, including misclassified data. Alternatively, the training of the regression CNN model can be completely separated from the classification model; that is, the entire 6400 parametric finite element analysis cases involving pinholes can be employed to train the regression model. In this study, the correctly classified dataset, which consists of 6395 cases and is almost identical to the complete dataset, is employed to train and validate the regression CNN model.

4.2.2. Proposed Structure

The proposed regression CNN model in the present study comprises a total of 23 layers: layers 1 to 16 are dedicated to feature extraction, while layers 17 to 23 are focused on regression. Layer 23 is the output layer, which includes four components representing the predicted r_p , d_p/t , h , and ϕ values, respectively. The detailed CNN regression model layer information is presented in Table 4. It is noted that, in addition to the dropout layers introduced in Section 4.1.2, several batch normalization layers are incorporated within the typical CNN layers to prevent overfitting [46–48]. As shown in Table 4, the tuned hyperparameters in the feature extraction process of the regression model, such as the filter size of the convolution layer, the pooling size, and the dropout rate, are the same as those in the classification CNN model, while the employed dropout rate in the regression process of the regression CNN model is 0.5. Figure 10 depicts the architecture of the developed regression model with the dimensions of different layers. Note that the dropout and batch normalization layers are excluded from the diagram as they do not impact the data dimensionality. The color scheme utilized to distinguish various layers in Figure 10 is consistent with those utilized in Figure 7.

Table 4. CNN regression model layer information.

Part	Layer No.	Layer Name	Parameters
Feature extraction	1	Convolution	64 filters with size equals to (5×5)
	2	Batch normalization	
	3	Maxpooling	Pooling size equals to (2×2)
	4	Dropout	Rate = 0.2
	5	Convolution	128 filters with size equals to (5×5)
	6	Batch normalization	
	7	Maxpooling	Pooling size equals to (2×2)
	8	Dropout	Rate = 0.2
	9	Convolution	256 filters with size equals to (5×5)
	10	Batch normalization	
	11	Maxpooling	Pooling size equals to (2×2)
	12	Dropout	Rate = 0.2
	13	Convolution	512 filters with size equals to (5×5)
	14	Batch normalization	
	15	Maxpooling	Pooling size equals to (2×2)
	16	Dropout	Rate = 0.2
Regression	17	Flatten layer	Unit number equals to 51,200
	18	Dropout	Rate = 0.5
	19	Fully connected layer	Unit number equals to 64
	20	Fully connected layer	Unit number equals to 128
	21	Fully connected layer	Unit number equals to 256
	22	Fully connected layer	Unit number equals to 512
	23	Output 1	Unit number equals to 1
		Output 2	Unit number equals to 1
		Output 3	Unit number equals to 1
		Output 4	Unit number equals to 1

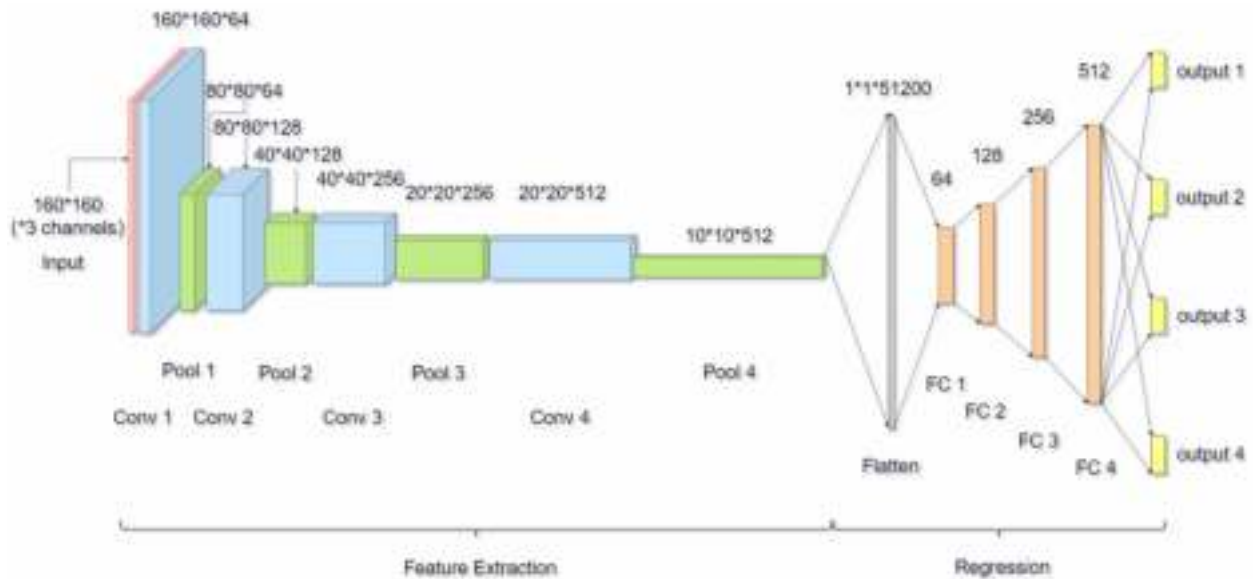


Figure 10. Proposed CNN regression model structure.

4.2.3. Results

The coefficient of determination (i.e., R^2) is used to measure the accuracy of each parameter prediction (i.e., r_p , d_p/t , h , and ϕ), defined by the following equation.

$$R^2 = 1 - \frac{\sum_{i=1}^N (Y_{true,i} - Y_{pred,i})^2}{\sum_{i=1}^N (Y_{true,i} - \bar{Y}_{true})^2} \quad (4)$$

where $Y_{true,i}$ and $Y_{pred,i}$ represent the true and predicted values, respectively, of each predicted parameter for the i th ($i = 1, 2, \dots, N$) point in a given dataset containing N data points, and $\bar{Y}_{true}$ denotes the true mean value of N data points associated with a predicted parameter. Table 5 summarizes the R^2 values for the four parameters in the test dataset included in the regression model. The R^2 values are also plotted in a so-called radar chart in Figure 11. The true values of the four parameters r_p , d_p/t , h , and ϕ compared with the estimated values by the CNN for the corruptions in the test set are depicted in Figure 12. It is evident from Table 5 that the CNN model's predictions of the corrosion location parameters (i.e., h and ϕ) show strong agreement with the actual values, as reflected by the R^2 values of 1.0 for all four corrosion categories. In terms of the size predictions (i.e., r_p and d_p/t), the model is more accurate for internal corruptions (P_{in} and PIC_{in}) than for external corruptions (P_{ex} and PIC_{ex}). These results also indicate that the performance of the model is better for cases containing pinholes only than for cases containing PIC. This is expected as PIC introduces additional complexity, including the presence of general corrosion, which may impact the model's accuracy. Overall, the obtained R^2 values demonstrate the CNN model's accuracy in predicting the size parameters, with a higher accuracy observed for corruptions on the internal surface and pinholes. However, note that although the predictions for r_p and d_p/t are, in general, accurate, Figure 12 reveals relatively large error bands in both r_p and d_p/t predictions. For instance, the predicted values of r_p range from 1.14 mm to 3.45 mm for $(r_p)_{true}$ equal to 2 mm. The large error bands may be explained by the small dimensions of the pinhole corrosion and the limited spatial resolution of the model. Furthermore, the presence of composite corrosion features can introduce additional complexities and uncertainties, leading to larger error bands in the predictions. Therefore, further investigation is required to explore the factors contributing to the reduction of regression errors for pinhole corruptions.

Table 5. Results of the regression CNN model without noise.

Category No.	Category	Cases			Test R^2			
		Total	Train	Test	r_p	d_p/t	h	ϕ
0	P _{in}	1600	1280	320	0.91	0.95	1.00	1.00
1	P _{ex}	1598	1277	321	0.88	0.92	1.00	1.00
2	PIC _{in}	1599	1280	319	0.89	0.85	1.00	1.00
3	PIC _{ex}	1598	1279	319	0.86	0.77	1.00	1.00
Overall		6395	5116	1279	0.89	0.91	1.00	1.00

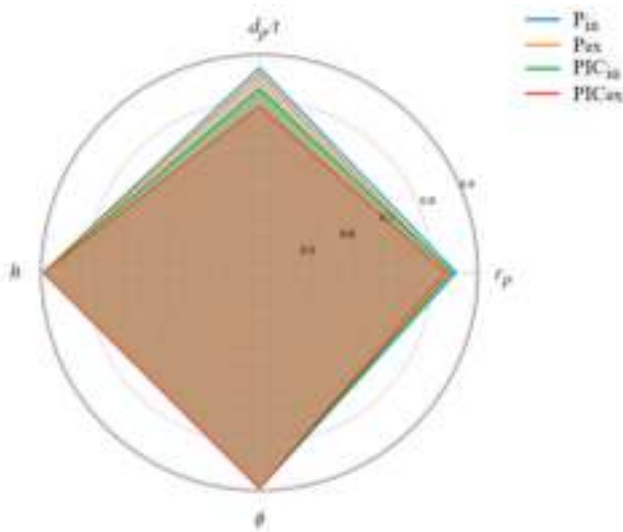


Figure 11. Radar chart for R^2 of metrics (r_p , d_p/t , h , and ϕ) for P_{in}, P_{ex}, PIC_{in}, and PIC_{ex} corrosions in the test dataset.

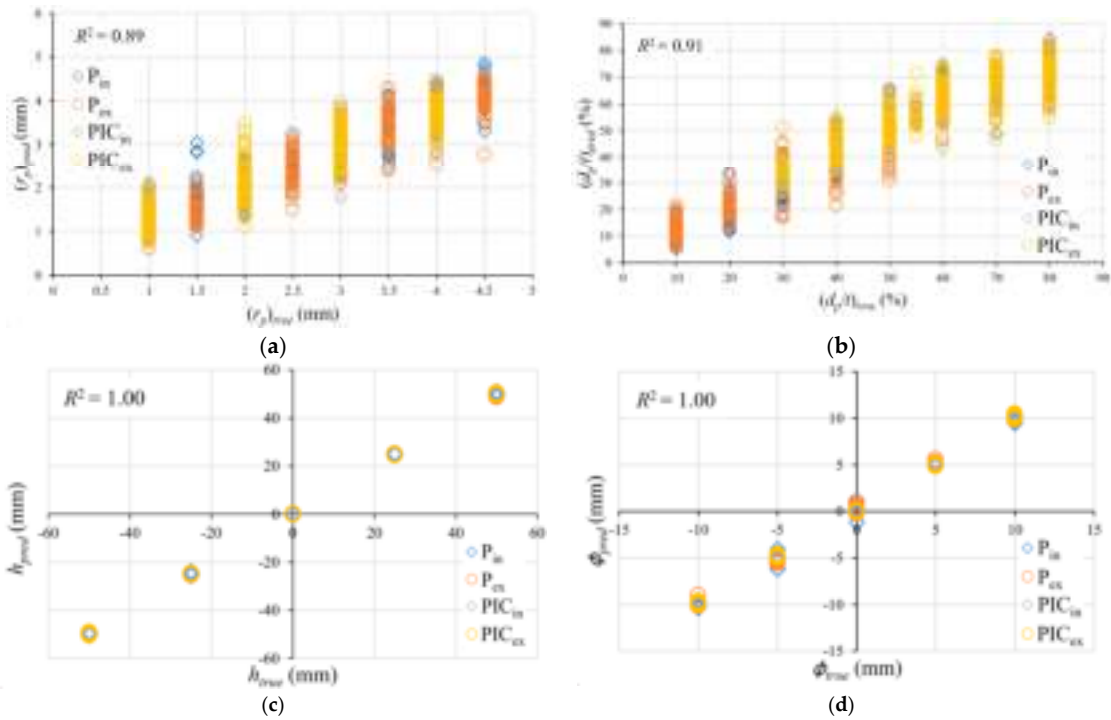


Figure 12. Comparison of true and predicted values of r_p , d_p/t , h , and ϕ for the anomalies in the regression test dataset without noise.

4.2.4. Influence of Noise

Figure 13 compares the results of the regression CNN model for noisy input (SNR = 20) with those of the noise-free scenario described in Section 4.2.3. With the presence of noise, the R^2 values for predicting r_p and d_p/t decrease somewhat compared to the noise-free scenario. The decrease in R^2 values is more prominent for P_{ex} and PIC_{ex} cases, indicating that the model is more affected by noise when predicting the sizes of external corrosion. Despite the decrease in the accuracy in predicting the corrosion sizes, the model maintains $R^2 = 1$ for the location parameters (h and ϕ) across all corrosion types except for P_{ex} , indicating its robustness in capturing the spatial information even in the presence of noise.

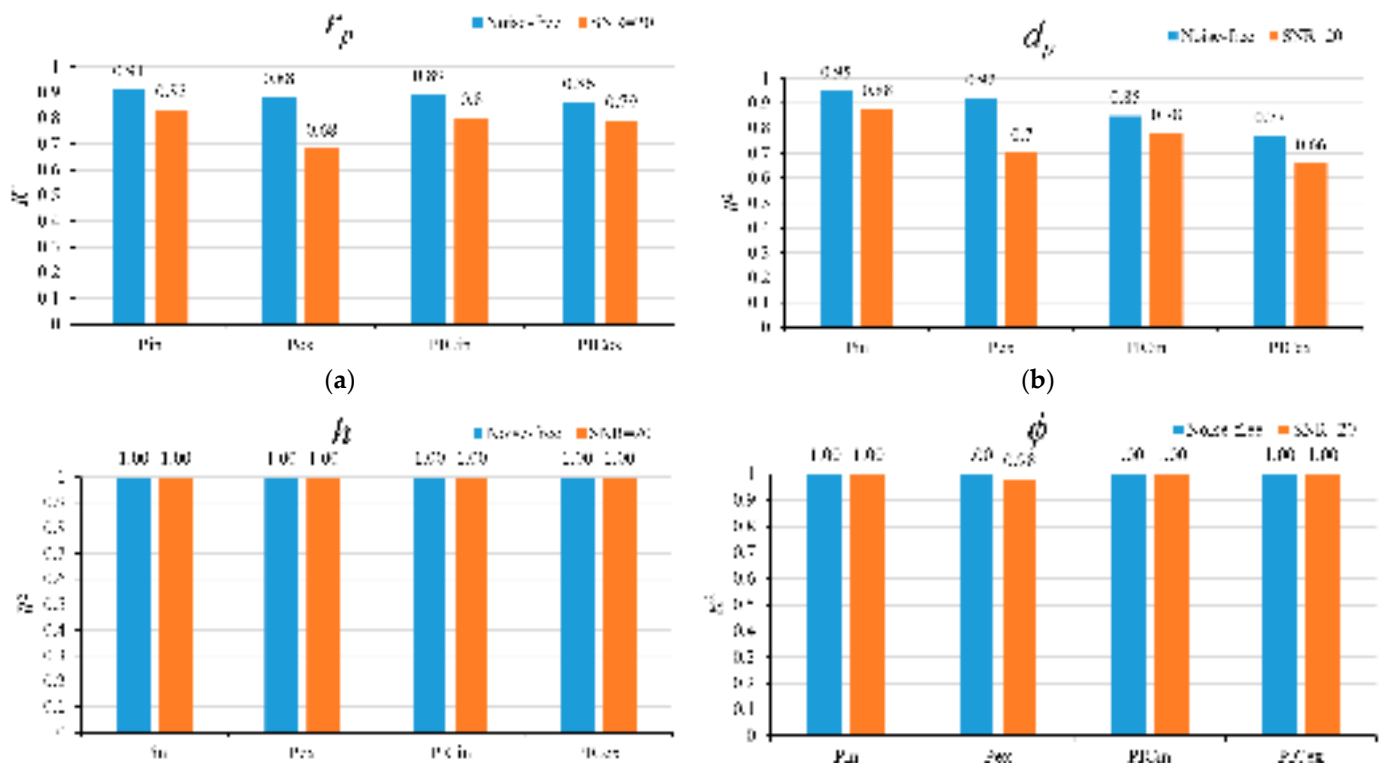


Figure 13. Comparison of R^2 between noise-free and SNR = 20 scenarios for different corrosion types.

Finally, it is important to point out that the CNN classification and regression models developed in this study are specifically tuned to certain pipeline and MFL tool attributes, such as the pipe wall thickness, magnet properties, lift-off values, and defect shapes. Therefore, further investigations are needed to determine how well the CNN model performs with various pipe wall thicknesses and different MFL tool parameters. Further studies are also needed to take into account the movement of the MFL tool into the prediction models.

5. Conclusions

This paper reports a novel study that focuses on the classification and sizing of pinhole corruptions on pipelines using deep learning algorithms. We propose a CNN classification model to classify six different types of corruptions (i.e., G_{in} , G_{ex} , P_{in} , P_{ex} , PIC_{in} , and PIC_{ex}) on pipelines and a CNN regression model to estimate the sizes and location of the pinhole defects based on MFL signals generated using magneto-static FEA. Extensive 3D parametric FEA cases involving box-shaped general corruptions and cylinder-shaped pinholes are used to simulate the 3D MFL signals by varying the defect depth, length, width, and longitudinal and circumferential locations. The CNN classification and regression models are then trained and validated using the simulated MFL signals.

The proposed classification model is shown to have excellent accuracy in classifying six types of corrosion defects with a misprediction rate of 0.34% ($33/9600 = 0.34\%$). The

proposed regression model is highly accurate in predicting the location of the pinhole and achieves good accuracy in predicting the depth and diameter of the pinhole. Through a comparative analysis of the CNN regression model with noise-free and noisy signals (SNR = 20) as input, it is observed that the noise impact on the predictive accuracy of the regression model is moderate. This study demonstrates the application of deep learning algorithms to facilitate the integrity management of pipelines containing complex-shaped corrosion defects.

Author Contributions: Y.S.: Investigation, Visualization, Formal analysis, Methodology, Writing—original draft. W.Z.: Funding acquisition, Supervision, Conceptualization, Methodology, Writing—review and editing. All authors have read and agreed to the published version of the manuscript.

Funding: The authors acknowledge the financial support received from the Natural Sciences and Engineering Research Council of Canada (NSERC) (Grant No. RGPIN-2019-05160) and the Faculty of Engineering of the University of Western Ontario.

Data Availability Statement: Data are available on request from the authors.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Zhou, Q.; Wu, W.; Liu, D.; Li, K.; Qiao, Q. Estimation of corrosion failure likelihood of oil and gas pipeline based on fuzzy logic approach. *Eng. Fail. Anal.* **2016**, *70*, 48–55. [CrossRef]
2. Murphy, J.F. Nightmare pipeline failures, fantasy planning, black swans, and integrity management—A review. *Process Saf. Prog.* **2015**, *34*, 207. [CrossRef]
3. Shen, Y.; Zhou, W. A comparison of onshore oil and gas transmission pipeline incident statistics in Canada and the United States. *Int. J. Crit. Infrastruct. Prot.* **2024**, *45*, 100679. [CrossRef]
4. Vanaei, H.R.; Eslami, A.; Egbewande, A. A review on pipeline corrosion, in-line inspection (ILI), and corrosion growth rate models. *Int. J. Press. Vessels Pip.* **2017**, *149*, 43–54. [CrossRef]
5. Song, H.; Yang, L.; Liu, G.; Tian, G.; Ona, D.I.; Song, Y.; Li, S. Comparative analysis of in-line inspection equipments and technologies. *IOP Conf. Ser. Mater. Sci. Eng.* **2018**, *382*, 032021. [CrossRef]
6. Pipeline Operators Forum. Specifications and Requirements for In-Line Inspection of Pipelines. Version 2021. Available online: <https://pipelineoperators.org/documents> (accessed on 18 May 2022).
7. Sutherland, J.; Bluck, M.; Pearce, J.; Quick, E. Validation of latest generation MFL in-line inspection technology leads to improved detection and sizing specification for pinholes, pitting, axial grooving and axial slotting. In Proceedings of the ASME 2010 8th International Pipeline Conference, Calgary, AB, Canada, 27 September–1 October 2010.
8. Peng, X.; Anyaoha, U.; Liu, Z.; Tsukada, K. Analysis of magnetic-flux leakage (MFL) data for pipeline corrosion assessment. *IEEE Trans. Magn.* **2020**, *56*, 1–15. [CrossRef]
9. Han, W.; Yang, P.; Xia, F.; Xue, Y. Magnetic flux leakage signal inversion of corrosive flaws based on modified genetic local search algorithm. *J. Shanghai Jiaotong Univ. (Sci.)* **2009**, *14*, 168–172. [CrossRef]
10. Priewald, R.H.; Magele, C.; Ledger, P.D.; Pearson, N.R.; Mason, J.S.D. Fast magnetic flux leakage signal inversion for the reconstruction of arbitrary defect profiles in steel using finite elements. *IEEE Trans. Magn.* **2012**, *49*, 506–516. [CrossRef]
11. Hwang, K.; Mandayam, S.; Udpa, S.S. Characterization of gas pipeline inspection signals using wavelet basis function neural networks. *NDT E Int.* **2000**, *33*, 531–545. [CrossRef]
12. Ramuhalli, P.; Udpa, L.; Udpa, S.S. Neural network based inversion algorithms in magnetic flux leakage nondestructive evaluation. *J. Appl. Phys.* **2003**, *93*, 82748276. [CrossRef]
13. Han, W.H.; Que, P.W. 2-D defect reconstruction from MFL signals based on genetic optimization algorithm. In Proceedings of the IEEE 2005 International Conference on Industrial Technology, Hong Kong, China, 14–17 December 2005.
14. Kandroodi, M.R.; Araabi, B.N.; Bassiri, M.M.; Ahmadabadi, M.N. Estimation of depth and length of defects from magnetic flux leakage measurements: Verification with simulations, experiments, and pigging data. *IEEE Trans. Magn.* **2016**, *53*, 1–10. [CrossRef]
15. Feng, J.; Li, F.; Lu, S.; Liu, J. Fast reconstruction of defect profiles from magnetic flux leakage measurements using a RBFNN based error adjustment methodology. *IET Sci. Meas. Technol.* **2017**, *11*, 262–269. [CrossRef]
16. Yao, G.; Lei, T.; Zhong, J. A review of convolutional-neural-network-based action recognition. *Pattern Recognit. Lett.* **2019**, *118*, 14–22. [CrossRef]
17. Taye, M.M. Theoretical understanding of convolutional neural network: Concepts, architectures, applications, future directions. *Computation* **2023**, *11*, 52. [CrossRef]
18. Lu, S.; Feng, J.; Zhang, H.; Liu, J.; Wu, Z. An estimation method of defect size from MFL image using visual transformation convolutional neural network. *IEEE Trans. Ind. Inform.* **2018**, *15*, 213–224. [CrossRef]

19. Shen, Y.; Zhou, W. Development of a convolutional neural network model to predict the size and location of corrosion defects on pipelines based on magnetic flux leakage signals. *Int. J. Press. Vessels Pip.* **2023**, *207*, 105–123. [\[CrossRef\]](#)
20. Wang, H.A.; Chen, G. Defect size estimation method for magnetic flux leakage signals using convolutional neural networks. *Insight* **2020**, *62*, 86–91. [\[CrossRef\]](#)
21. EGIG. *10th Report of the European Gas Pipeline Incident Data Group (Period 1970–2016)*; EGIG: Groningen, The Netherlands, 2018; Doc. number VA 17.R.0395.
22. Zhang, H.; Sha, S.; Willis, C.; Qingshan, F.; Chen, P. Feasibility study of pinhole inspection via magnetic flux leakage and hydrostatic testing in oil & gas pipelines. *IOP Conf. Ser. Mater. Sci. Eng.* **2021**, *1043*, 022053.
23. Feng, Q.; Yan, B.; Chen, P.; Shirazi, S.A. Failure analysis and simulation model of pinhole corrosion of the refined oil pipeline. *Eng. Fail. Anal.* **2019**, *106*, 1–28. [\[CrossRef\]](#)
24. Subramanian, C. Localized pitting corrosion of API 5L grade A pipe used in industrial fire water piping applications. *Eng. Fail. Anal.* **2018**, *92*, 405–417. [\[CrossRef\]](#)
25. Askari, M.; Aliofkhazraei, M.; Afroukhteh, S. A comprehensive review on internal corrosion and cracking of oil and gas pipelines. *J. Nat. Gas Sci. Eng.* **2019**, *71*, 102971. [\[CrossRef\]](#)
26. Kadhim, K.N.; Al-Rufaye, A.H.R. The effects of uniform transverse magnetic field on local flow and velocity profile. *Int. J. Civ. Eng. Technol.* **2016**, *7*, 140–151.
27. Ji, F.; Wang, C.; Sun, S.; Wang, W. Application of 3-D FEM in the simulation analysis for MFL signals. *Insight* **2009**, *51*, 32–35. [\[CrossRef\]](#)
28. Bubenik, T. *Electromagnetic Methods for Detecting Corrosion in Underground Pipelines: Magnetic Flux Leakage (MFL)*; Underground Pipeline Corrosion; Woodhead Publishing: Cambridgeshire, UK, 2014; pp. 215–226.
29. Shi, Y.; Zhang, C.; Li, R.; Cai, M.; Jia, G. Theory and application of magnetic flux leakage pipeline detection. *Sensors* **2015**, *15*, 31036–31055. [\[CrossRef\]](#)
30. Zhang, Y.; Ye, Z.; Wang, C. A fast method for rectangular crack sizes reconstruction in magnetic flux leakage testing. *NDT E Int.* **2009**, *42*, 369–375. [\[CrossRef\]](#)
31. Walker, J. *In-Line Inspection of Pipelines: Advanced Technologies for Economic and Safe Operation of Oil and Gas Pipelines*; Verlag Moderne Industrie: Landsberg am Lech, Germany, 2010.
32. Liu, Y.; Gao, X.; Wang, Y.; Yang, X. Sensitive parameters' optimization of the permanent magnet supporting mechanism. *J. Mech. Sci. Technol.* **2014**, *28*, 2707–2714. [\[CrossRef\]](#)
33. Yang, L.; Zhang, G.; Liu, G.; Gao, S. Effect of lift-off on pipeline magnetic flux leakage inspection. In Proceedings of the 2008 17th World Conference on Nondestructive Testing, Shanghai, China, 25–28 October 2008.
34. Ireland, R.C.; Torres, C.R. Finite element modelling of a circumferential magnetizer. *Sens. Actuators A Phys.* **2006**, *129*, 197–202. [\[CrossRef\]](#)
35. Azizzadeh, T.; Safizadeh, M.S. Three-dimensional finite element and experimental simulation of magnetic flux leakage-type NDT for detection of pitting corrosions. In Proceedings of the 2017 4th Iranian International NDT Conference (IRNDT), Tehran, Iran, 21–22 February 2017.
36. Chen, L.; Li, X.; Qin, G.; Lu, Q. Signal processing of magnetic flux leakage surface flaw inspect in pipeline steel. *Russ. J. Nondestruct. Test.* **2008**, *44*, 859–867. [\[CrossRef\]](#)
37. Piao, G.; Guo, J.; Hu, T.; Leung, H. The effect of motion-induced eddy current on high-speed magnetic flux leakage (MFL) inspection for thick-wall steel pipe. *Res. Nondestruct. Eval.* **2020**, *31*, 48–67. [\[CrossRef\]](#)
38. Li, F.; Feng, J.; Zhang, H.; Liu, J.; Lu, S.; Ma, D. Quick reconstruction of arbitrary pipeline defect profiles from MFL measurements employing modified harmony search algorithm. *IEEE Trans. Instrum. Meas.* **2018**, *67*, 9. [\[CrossRef\]](#)
39. Fan, X.; Dai, M.; Liu, C.; Wu, F.; Yan, X.; Feng, Y.; Feng, Y.; Su, B. Effect of image noise on the classification of skin lesions using deep convolutional neural networks. *Tsinghua Sci. Technol.* **2019**, *25*, 425–434. [\[CrossRef\]](#)
40. Acharya, U.R.; Oh, S.L.; Hagiwara, Y.; Tan, J.H.; Adam, M.; Gertych, A.; San Tan, R. A deep convolutional neural network model to classify heartbeats. *Comput. Boil. Med.* **2017**, *89*, 389–396. [\[CrossRef\]](#) [\[PubMed\]](#)
41. Tesfai, H.; Saleh, H.; Al-Qutayri, M.; Mohammad, M.B.; Tekeste, T.; Khandoker, A.; Mohammad, B. Lightweight shufflenet based cnn for arrhythmia classification. *IEEE Access.* **2022**, *10*, 111842–111854. [\[CrossRef\]](#)
42. West, N.E.; O'shea, T. Deep architectures for modulation recognition. In Proceedings of the 2017 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN), Baltimore, MD, USA, 6–9 March 2017.
43. Baranwal, S.; Khandelwal, S.; Arora, A. Deep learning convolutional neural network for apple leaves disease detection. In Proceedings of the 2019 International Conference on Sustainable Computing in Science, Technology and Management (SUSCOM), Amity University Rajasthan, Jaipur, India, 26–28 February 2019.
44. Virupakshappa, K.; Marino, M.; Oruklu, E. A multi-resolution convolutional neural network architecture for ultrasonic flaw detection. In Proceedings of the 2018 IEEE International Ultrasonics Symposium (IUS), Kobe, Japan, 22–25 October 2018.
45. Bilmes, J. Underfitting and Overfitting in Machine Learning. UW ECE Course Notes. 2020. Available online: https://people.ece.uw.edu/bilmes/classes/ee511/ee511_spring_2020/overfitting_underfitting.pdf (accessed on 20 March 2023).
46. Ioffe, S.; Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of the 2015 International Conference on Machine Learning, Lille, France, 6–11 July 2015.

47. Thakkar, V.; Tewary, S.; Chakraborty, C. Batch normalization in convolutional neural networks—A comparative study with CIFAR-10 data. In Proceedings of the IEEE 2018 Fifth International Conference on Emerging Applications of Information Technology (EAIT), IIST Shibpur, West Bengal, India, 12–13 January 2018.
48. Bjorck, N.; Gomes, C.P.; Selman, B.; Weinberger, K.Q. Understanding batch normalization. *Adv. Neural Inf. Process. Syst.* **2018**, *31*, 1–12.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.



Article

The Parallel Machine Scheduling Problem with Different Speeds and Release Times in the Ore Hauling Operation

Luis Tarazona-Torres , **Ciro Amaya ***, Alvaro Paipilla, Camilo Gomez and David Alvarez-Martinez

Industrial Engineering Department, University of Los Andes, Cra 1 N18A12, Bogota 111711, Colombia; le.tarazona@uniandes.edu.co (L.T.-T.); a.paipilla@uniandes.edu.co (A.P.); gomez.ch@uniandes.edu.co (C.G.); d.alvarezm@uniandes.edu.co (D.A.-M.)

* Correspondence: ca.amaya@uniandes.edu.co

Abstract: Ore hauling operations are crucial within the mining industry as they supply essential minerals to production plants. Conducted with sophisticated and high-cost operational equipment, these operations demand meticulous planning to ensure that production targets are met while optimizing equipment utilization. In this study, we present an algorithm to determine the minimum amount of hauling equipment required to meet the ore transport target. To achieve this, a mathematical model has been developed, considering it as a parallel machine scheduling problem with different speeds and release times, focusing on minimizing both the completion time and the costs associated with equipment use. Additionally, another algorithm was developed to allow the tactical evaluation of these two variables. These procedures and the model contribute significantly to decision-makers by providing a systematic approach to resource allocation, ensuring that loading and hauling equipment are utilized to their fullest potentials while adhering to budgetary constraints and operational schedules. This approach optimizes resource usage and improves operational efficiency, facilitating continuous improvement in mining operations.



Citation: Tarazona-Torres, L.; Amaya, C.; Paipilla, A.; Gomez, C.; Alvarez-Martinez, D. The Parallel Machine Scheduling Problem with Different Speeds and Release Times in the Ore Hauling Operation. *Algorithms* **2024**, *17*, 348. <https://doi.org/10.3390/a17080348>

Academic Editors: Alexandre Dolgui, David Lemoine, Maria I. Restrepo and Frank Werner

Received: 2 May 2024

Revised: 24 July 2024

Accepted: 30 July 2024

Published: 8 August 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

Keywords: parallel machine scheduling problem; ore hauling equipment; mathematical model; mining industry

1. Introduction

One of the most essential activities in mining operations is ore hauling; this process involves moving fragmented rock from the blasting site using loading and hauling equipment [1]. The ore rocks are then transported to the crushing process to reduce their size before being sent to the recovery plant, where valuable metals such as copper, gold, molybdenum, and aluminum, among others, are extracted [2–4]. Figure 1 illustrates the schematic of the described process. Poor planning of ore hauling operations can decrease productivity levels, affect the achievement of production targets within the established timelines, and consequently impact operational costs. Several factors must be considered to effectively plan ore hauling operations, such as the amount of available loading and hauling equipment and their scheduled maintenance [5]. It is also essential to account for the established production targets, working hours, and topographic levels. Other important aspects include the ore grade, the number of available loading ramps for loading operations, and the environmental impact [6].

Mining companies have a defined amount of loading and hauling equipment allocated based on the production targets set for a specific period. It is crucial to determine the necessary amount of this equipment to efficiently transport ore rocks to control associated costs, which can be substantial depending on the time required to meet these targets [7]. Knowing the minimum amount of equipment needed optimizes operations and facilitates tactical analysis. This analysis enables better decision-making in the face of contingencies, ensuring the achievement of production targets at the lowest possible cost.

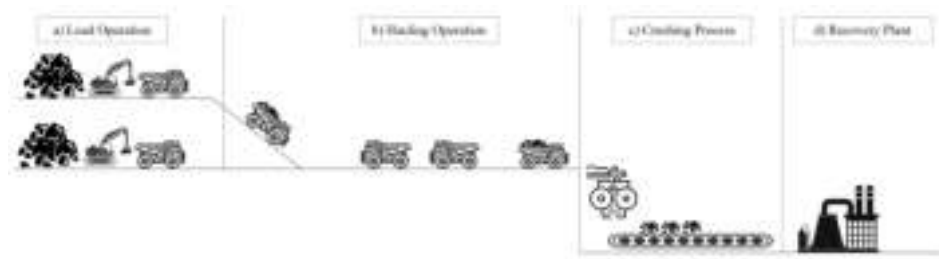


Figure 1. Schematic of the ore hauling operation.

As shown in Figure 1, the ore hauling process involves assigning hauling equipment to loading equipment. These do not necessarily operate at the same loading speed because some equipment have different loading capacities. The loading is performed simultaneously, and is described as a system of parallel machines with different speeds. Since the number of hauling equipment is finite, they are used as many times as necessary to meet the production target. However, they are unavailable while transporting the ore to the crushing process, having a “release time” before they are available again to be reloaded. Thus, this scenario can be adapted to a problem of parallel machines with different speeds and release times.

The novelty of this work lies in incorporating different speeds and release times into the mathematical model for parallel machine scheduling in mineral hauling operations (PMSPOHO). These features allow for a more realistic and accurate representation of operational conditions in the mining environment. The speed of the loading equipment plays a crucial role in this model. Variations in loading speeds can significantly impact the efficiency of the ore hauling operation. By considering different loading speeds, the model can more precisely allocate resources to match the dynamic nature of mining operations. This ensures that faster-loading equipment is utilized to its maximum potential while slower equipment is assigned tasks that better suit its capabilities, optimizing the overall workflow.

Incorporating the speed of the loading equipment into the scheduling algorithm maximizes equipment use and minimizes idle time for the hauling equipment. This enhances operational efficiency by reducing delays and ensuring a continuous flow of materials. The developed mathematical model addresses these critical variables, optimizing the allocation of hauling equipment, maximizing their use, and reducing idle time. This approach ensures greater operational efficiency by aligning equipment capabilities with the demands of the mining process. The strategic allocation of resources, guided by the speed and availability of each loading machine, contributes to a more streamlined and cost-effective operation, highlighting the importance of these variables in achieving optimal performance.

The mathematical model contributes significantly to several vital aspects. It facilitates more efficient planning and scheduling of hauling operations, allowing for dynamic adjustments based on changing conditions. Additionally, it enables a comparative analysis between costs and completion times, optimizing economic and productivity decisions. By reducing downtime and improving resource utilization, the model effectively addresses the logistical and financial challenges in mining operations and adapts to various operational scenarios, increasing its applicability and relevance in the sector. The remainder of this paper is structured as follows: Section 2 presents the related work. Section 3 details the methodology. Computational experiments and results analysis are discussed in Section 4. Finally, conclusions drawn from the study are presented in Section 5.

2. Related Works

Table 1 presents an overview of related works addressing the problem. It details the characteristics considered in our area of interest, such as release times (r_i) and different speeds (Q_m). The objective functions used are also analyzed, such as completion time (C_{max}) and costs (k), among others. Additionally, the solution methods employed are examined and categorized as exact or approximate, along with the number of machines utilized and

the approaches adopted in each instance. This comparison allows for a visualization of the various methodologies and techniques implemented in the literature, highlighting their differences and similarities concerning the current analysis.

The characteristic r_i has yet to be sufficiently explored, particularly in mining research where the availability of hauling equipment is crucial, as shown by [8,9]. In contrast, studying Q_m has garnered more academic attention, though the focus largely remains on identical machines. Regarding the objective function, studies have generally centered on minimizing variables such as k and C_{max} in isolation. Meza [2] exemplifies a more complex integration, which incorporates a time and cost analysis of loading and hauling equipment through a simulation model. Hong and Lin [10] focused on minimizing maximum lateness (L_{max}), while Koryagin and Voronov [11] and Olishevych et al. [12] considered meeting P as essential for maintaining operational continuity in production plants.

In comparison, our work significantly expands the level of analysis by including both Q_m and r_i , allowing us to model the production environments more realistically, especially in ore hauling operations. This is particularly relevant in configurations where the number of machines exceeds five, a scenario in which many studies prefer approximate methods due to their computational efficiency. Additionally, our approach includes an iterative study that enables tactical-level analysis to define the optimal utilization of equipment, complemented by robust analytical support for decision-making to maximize operational effectiveness. Our approach also incorporates a complexity analysis that has yet to be explored [10,13,14]. Thus, our work addresses a significant gap in the literature and provides critical insights that facilitate strategic decisions and enhance operational effectiveness in the mining industry.

Table 1. Related works on the parallel machine scheduling problem.

Reference	Characteristics		Objective Function			Methods		Num. Machines		Complexity Analysis	Approaches
	Q_m	r_i	C_{max}	k	Other	Exact	Approximate	$m \leq 5$	$m \geq 6$		
[2]	x		x	x			x		x		FIFO simulation model
[15]			x			x		x			Branch-and-price algorithm with single server
[16]			x			x	x	x	x		MIP formulation and hybrid heuristic algorithm with single server
[17]			x			x	x	x			MIP formulation and Tabu-Search with single server
[18]			x			x		x			MIP formulation with non-identical job sizes
[19]	x			x		x		x			MIP formulation
[20]	x			x		x		x			MILP formulation
[21]	x			x		x	x	x	x		MILP formulation and Insertion Heuristic under time-of-use tariffs
[22]	x			x		x		x			MILP formulation under time-of-use tariffs
[8]		x	x			x	x	x	x		MILP formulation and GA algorithm
[11]	x				P		x	x			Heuristic algorithm
[23]			x			x		x			MIP formulation with setup time and single server
[10]			x		L_{max}		x	x		x	Dynamic constructive algorithms
[24]			x			x	x				Constraint programming model under unavailability constraints and modify LPT
[13]			x			x	x	x		x	MIP formulation and Tabu-Search

Table 1. Cont.

Reference	Characteristics		Objective Function			Methods		Num. Machines		Complexity Analysis	Approaches
	Q_m	r_i	C_{max}	k	Other	Exact	Approximate	$m \leq 5$	$m \geq 6$		
[25]				x		x			x		MIP formulation
[14]			x			x		x	x	x	MIP formulation with independent jobs
[26]	x		x			x		x			MIP formulation with function constrain
[12]	x				P	x		x			MIP formulation
[27]				x			x	x			Monte Carlo simulation model
[9]		x	x				x	x			VNS and ACO algorithms
[28]	x		x			x	x		x		MIP and TSP algorithm
[29]			x			x		x			Constraint programming model
[30]				x		x	x	x			MIP formulation and MOBSO algorithm
[31]	x			x		x	x	x			MILP formulation with time-of-use and SPT, MDPC, MDEC
[32]	x		x			x	x		x		MIP formulation and GA-GWO hybrid algorithm
Our paper	x	x	x	x		x		x	x	x	Iterative MIP formulation

Considering related studies on the optimization of the ore hauling operation, it was found that Meza [2] developed a model for the application of simulation concepts of a surface mining operation loading and hauling system, considering the type of loading and hauling equipment, arrival time, waiting time, positioning time, loading time, going time, setting time, unloading time, and return time, in addition to the number of trips as a production target, and considering a FIFO system for loading and unloading. On the other hand, Vasquez et al. [3] performed an integer programming model to minimize the total working time of a low-profile loader fleet in an underground mine, in addition to developing a polynomial time optimal algorithm integrated into the decision-making process, obtaining results quite close to the optimal total time. Eivazy and Askari-Nasab [33] developed a multi-destination mixed integer linear programming (MILP) model to minimize operations costs in surface mining; in the model, they considered stockpiles and mixing piles in addition to horizontal direction loading systems and ramp decision-making. Anjomshoa et al. [34] developed a simulation and mixed integer programming model (MIP) to optimize overtaking bays, sizing, and scheduling hauling equipment fleet in underground mining.

Moreover, Gligoric [35] developed an investigation on the overtaking bays for hauling equipment in underground mining, in which they sought to optimize the necessary amount of bays and the optimal location to minimize waiting times. Gonen et al. [36] conducted a study for underground mining where they evaluated annual production capacities with different hauling systems, considering unit hauling costs and mine depth. Eivazy and Askari-Nasab [37] presented a methodology based on production scheduling hierarchies for surface mining for the medium term, developing an MILP to minimize operating costs; the scheduling included stockpiles, processing plants, and dumps, as well as a selection of routes and ramps. Tom-Socarras [38] generated a tool for cost management based on a decision tree for material transportation in surface mining. Uribe [39] developed a study that analyzed the functionality and performance of semi-automatic operation in loading and hauling equipment used in underground mining. Więcek [40] proposed a suitable method for controlling the loading and transporting of ore in a mine with a piles and

chambers system. Wiest [41] developed a scheduling model for surface mining, focusing on cost reduction.

Solomon [42] modeled ore loading and hauling systems in surface mining, developing a model that considered the excavation face, routes, equipment, and ore destinations. Gaspar and Jhasmin [43] analyzed utilization and availability indicators for ore hauling cost optimization. Li et al. [44], in their research for underground mining, developed a bee swarm algorithm to maximize the total revenue during a planning period; in addition, they created a genetic algorithm for the planning for hauling equipment dispatching to minimize the waiting time of this equipment. Elijah [45] developed a mathematical model for calculating the arrival and service rate of different amounts of hauling equipment involved in material transportation. Huisa [46] improved the use of loading and hauling equipment to reduce the cost of these operations.

Choi et al. [47] developed an intelligent non-supervisory system to predict the performance of the hauling equipment system in surface mining using a combination of different optimization models. Li et al. [48] focused on a mathematical model for rescheduling operations considering the requirements of production spaces, operating environment, and production equipment wear, while seeking to obtain the maximum planning completion rate and the lowest ore grade fluctuation. Abolghasemian et al. [49] presented a multi-objective optimization in the surface mine system, seeking the maximum amount of extraction and minimizing the transportation time, considering the storage capacity, transportation equipment, and budget. Shamsi et al. [50] performed an IP with operational restrictions to schedule the transportation system in surface mining, considering the best net present value as an objective. Parichehreh et al. [51] addressed the energy-efficient unrelated parallel machine scheduling problem, incorporating job deterioration and the learning effects of operators, providing valuable insights into multi-objective optimization for complex production environments.

Our study introduces a novel and significant approach to optimizing ore loading and hauling operations by modeling a tactical problem using a scheduling model typically employed for operational decisions. This model accounts for parallel machines with different speeds, adding greater realism. Specifically, the nature of the problem is reflected in the dynamic availability of loading equipment, represented through dynamic release times, where jobs are not known in advance but are revealed as they are executed. This approach allows for deeper and more detailed analyses that are better suited to the complexities of mining production environments. Implementing this model can be integrated into tactical and operational strategies to make them more efficient and effective, supporting decision-making in mining companies by significantly improving operational efficiency, reducing costs, and increasing productivity. Furthermore, this study will contribute to both theoretical development and practical application in mining engineering, enhancing essential operations in the sector.

3. Methodology

This section outlines the methodology employed in this study. It is structured as follows: Section 3.1 defines the problem and sets the context for the research. Section 3.2 details the modeling assumptions and presents the mathematical formulation of the problem, including the objective functions and constraints. Section 3.3 describes the algorithms used for determining and evaluating the optimal amount of ore hauling equipment. This includes the procedure for determining the minimum amount of hauling equipment and the evaluation of the cost–time trade-offs. The following subsections systematically address the research objectives and provide a comprehensive understanding of the applied methods.

3.1. Problem Definition

Due to the critical importance of ore hauling for production, this research aims to determine the minimum amount of hauling equipment necessary to minimize both the costs associated with equipment usage and the completion time of the production target

within a defined time horizon. Ore hauling operations are pivotal for continuously supplying essential minerals to production plants. However, technical decisions regarding the required equipment are often made based on experience and under time pressure. While these decisions may be quick, they might only sometimes be optimal, potentially leading to increased costs or delays that impact plant production.

This research addresses the inefficiency and potential sub-optimality of traditional decision-making processes in determining the minimum amount of hauling equipment needed to realize the production target. This can result in either overestimating or underestimating hauling equipment, leading to unnecessary expenses or bottlenecks during ore transport. By developing a mathematical model that considers the parallel machine scheduling problem with different speeds and release times, we aim to provide a systematic and data-driven approach to optimizing equipment allocation.

A detailed analysis of how costs relate to completion time could reveal valuable scenarios, enabling more informed and appropriate decisions in the event of any contingency in the transportation operation. This model will allow decision-makers to balance the trade-offs between equipment costs and operational efficiency, ensuring that the right amount of equipment is deployed to meet production targets without incurring unnecessary expenses.

The problem can be conceptualized as a parallel machine scheduling problem with different speeds and release times. While the primary focus is a tactical decision to determine the minimum amount of hauling equipment and analyze costs related to completion time, an adaptation to this scheduling problem under an integer programming model is executed. According to the notation established by Graham [52], an operations scheduling problem is defined by three fields ($\alpha|\beta|\gamma$): α represents the system configuration, β the constraints and characteristics of the system, and γ the objective function. Using this framework and following the nomenclature described by Pinedo and Hadavi [53], the system configuration corresponds to parallel machines with different speeds (Q_m), where these machines are the loading equipment with varying processing times. A significant system characteristic includes the release times (r_i) incorporated within the hauling time. The objective function aims to minimize both costs and completion time. Figure 2 illustrates the schedule of the ore hauling operation conceptualized as the problem previously mentioned.

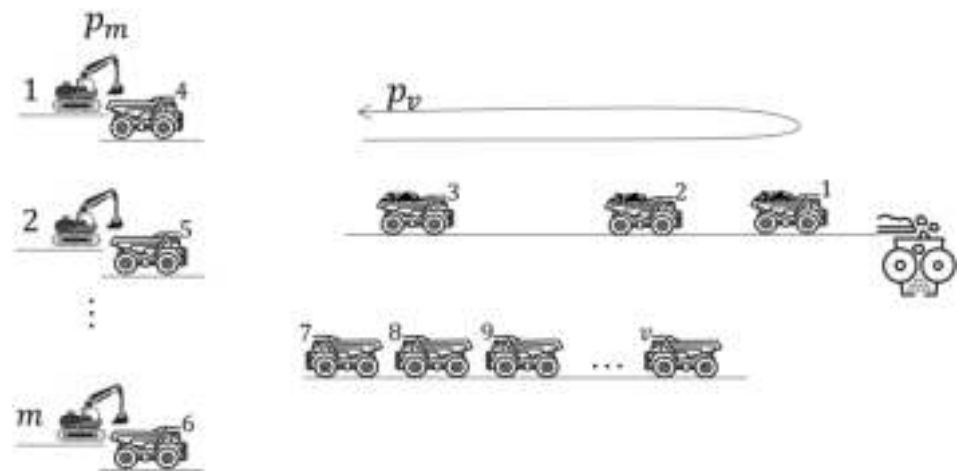


Figure 2. Schedule considering ore hauling operation.

3.2. Modeling

Model assumptions. The assumptions of the model are as follows, based on the work presented by Meza [2]:

- Only one ore unloading point was considered.
- The refueling time is negligible.
- The loading and hauling equipment is available at all times.
- The ore unloading waiting time is insignificant.

Notation. Table 2 provides the notation for the model proposed, showing the type and definition, respectively.

Table 2. Notation for the model proposed.

Var	Type	Definition
$\mathcal{M}$	Set	Set of loading equipment $\mathcal{M} = \{1, 2, \dots, m\}$.
$\mathcal{V}$	Set	Set of hauling equipment $\mathcal{V} = \{1, 2, \dots, v\}$.
$\mathcal{T}$	Set	Set of units of time: $\mathcal{T} = \{1, 2, \dots, h\}$.
p_j^l	Parameter	Loading time for loading equipment j .
p_i^h	Parameter	Hauling time for hauling equipment i .
P	Parameter	The production target is the total amount of trips to be made.
h	Parameter	Time horizon to complete the production target.
k_j^l	Parameter	Cost associated with use loading equipment.
k_i^h	Parameter	Cost associated with use hauling equipment.
x_{ijt}	Variable	Principal decision variable, takes the value of 1 if the hauling equipment i is using the loading equipment j at time t , 0 otherwise.
y_{jt}	Variable	The auxiliary decision variable takes the value of 1 if loading equipment j is using at time t , 0, otherwise.
z_{it}	Variable	The auxiliary decision variable takes the value of 1 if hauling equipment i is using at time t , 0, otherwise.
C_{max}	Variable	Completion time of the production target
k	Variable	Costs associated with equipment usage

Objective function. The first objective function (Equation (1)) seeks to minimize k associated to use equipment. The second objective function (Equation (2)) seeks to minimize C_{max} of all trips.

$$\min k \quad (1)$$

$$\min C_{max} \quad (2)$$

Constraints. With respect to constraints, the set of constraints in Equation (3) ensures the assignment of only one hauling vehicle i to one loading vehicle j at time t . The set of constraints in Equation (4) ensures only one loading equipment j per hauling equipment i at time t .

$$\sum_{i \in \mathcal{V}} x_{ijt} \leq 1 \quad \forall j \in \mathcal{M}, t \in \mathcal{T} \quad (3)$$

$$\sum_{j \in \mathcal{M}} x_{ijt} \leq 1 \quad \forall i \in \mathcal{V}, t \in \mathcal{T} \quad (4)$$

The set of constraints in Equations (5) and (6) ensures that the loading and hauling equipment are used for the required duration when assigned.

$$\sum_{\tau=t+1}^{t+p_j^l} y_{j\tau} \geq (p_j^l - 1) \cdot x_{ijt} \quad \forall i \in \mathcal{V}, j \in \mathcal{M}, t \in \mathcal{T} \setminus (t \leq h - p_j^l) \quad (5)$$

$$\sum_{\tau=t+1}^{t+p_i^h+p_j^l} z_{i\tau} \geq (p_i^h + p_j^l - 1) \cdot x_{ijt} \quad \forall i \in \mathcal{V}, j \in \mathcal{M}, t \in \mathcal{T} \setminus (t \leq h - p_i^h - p_j^l) \quad (6)$$

The set of constraints in Equations (7) and (8) ensures that loading and hauling equipment are not double-booked and are used only when available.

$$x_{ijt} \leq 1 - y_{jt} \quad \forall i \in \mathcal{V}, j \in \mathcal{M}, t \in \mathcal{T} \quad (7)$$

$$x_{ijt} \leq 1 - z_{it} \quad \forall i \in \mathcal{V}, j \in \mathcal{M}, t \in \mathcal{T} \quad (8)$$

The set of constraints in Equation (9) ensures C_{max} .

$$C_{max} \geq t \cdot x_{ijt} + p_i^h + p_j^l \quad \forall i \in \mathcal{V}, j \in \mathcal{M}, t \in \mathcal{T} \quad (9)$$

The set of constraints in Equation (10) ensures that P is met.

$$\sum_i^V \sum_j^M \sum_t^T x_{ijt} \geq P \quad \forall i \in \mathcal{V}, j \in \mathcal{M}, t \in \mathcal{T} \quad (10)$$

Equation (11) calculates k .

$$k = \sum_i^V \sum_t^T k_i^h \cdot z_{it} + \sum_j^M \sum_t^T k_j^l \cdot y_{jt} \quad \forall i \in \mathcal{V}, j \in \mathcal{M}, t \in \mathcal{T} \quad (11)$$

The set of constraints in Equations (12) and (13) ensures the nature of the variables.

$$x_{ijt}, y_{jt}, z_{it} \in \{0, 1\} \quad \forall i \in \mathcal{V}, j \in \mathcal{M}, t \in \mathcal{T} \quad (12)$$

$$C_{max}, k \geq 0 \quad (13)$$

The proposed model can be used in tactical and operational decision-making processes, as it accurately specifies both decisions. However, the complexity of this mixed-integer linear programming model in terms of the amount of integer variables is $i \cdot j \cdot t + j \cdot t + i \cdot j$, and it has two more continuous variables. If we analyze the number of constraints, we find a total of $7 \cdot i \cdot j \cdot t + j \cdot t + i \cdot j$. The fact that the model relies on binary vectors (to indicate truck start times and activity periods) inherently increases complexity but also opens the possibility for (i) the inclusion of valid inequalities to discard solutions that do not fit the structure of the problem or (ii) decomposition techniques, such as using combinatorial Benders' cuts, or adopting a rolling horizon framework [54], to leverage divide and conquer strategies; these are both part of ongoing research. One could consider reducing the periods, but this would prevent using these models in control schemes such as those provided by a digital twin, a technique widespread in mining operations management [55].

3.3. Algorithms for Determining and Evaluating the Optimal Amount of Ore Hauling Equipment

To incorporate the dual objectives of the model of minimizing both the completion time and cost, an iterative approach was utilized in which models are created using only one objective function. As shown in Algorithms 1 and 2, the iterative process allows for the exploration of the trade-off between completion time and costs, providing a set of solutions that balance both objectives.

Considering the tactical-level analysis, the following procedure was proposed to determine the optimal amount of hauling equipment, as detailed in Algorithm 1. The algorithm starts by defining the input data, which include $\mathcal{M}$, $\mathcal{V}$, $\mathcal{T}$, p_i^h , p_j^l , h , and P . Then, defining the output data, which include the minimum number of hauling equipment (v_{opt}), and the hauling equipment' assignment to loading equipment (*Schedule*) including C_{max} .

Then, assign v to the magnitude of $\mathcal{V}$, the total number of hauling equipment available (line 1). The variable *Continue* is set to true to initiate the loop (line 2). As long as it remains true, a model is created (considering Equations (2)–(13) of PMSPOHO model, input data and v), and a *Result* that minimizes the C_{max} is obtained (lines 3–5). A feasibility evaluation is conducted, where if the *Result* is feasible, v is reduced by one unit (attempting to solve the problem with fewer hauling equipment) (lines 6–8). If it is not feasible, v_{opt} is increased by one unit, and the loop terminates (lines 9–13). This algorithm returns v_{opt} and *Schedule* (line 14).

Algorithm 1 Optimal hauling equipment determinator.

Input: $\mathcal{M}, \mathcal{V}, \mathcal{T}, p_i^h, p_j^l, h, P$;
Output: v_{opt} : minimum hauling equipment, *Schedule*: hauling equipment' assignment to loading equipment including the C_{max} .

```

1:  $v \leftarrow |\mathcal{V}|$ 
2: Continue  $\leftarrow$  true
3: while Continue do
4:    $model \leftarrow \text{Createmodel}(\text{Equations}(2)–(13), \text{Input}, v)$ 
5:    $Result \leftarrow \text{Solve}(model, \text{Minimize})$ 
6:   if  $\text{IsFeasible}(Result)$  then
7:      $v \leftarrow v - 1$ 
8:      $Result, Schedule$ 
9:   else
10:     $v_{opt} \leftarrow v + 1$ 
11:    Continue  $\leftarrow$  false
12:   end if
13: end while
14: return  $v_{opt}, Schedule$ 

```

The following procedure was proposed for an analysis of the relationship between k and C_{max} , detailed in Algorithm 2. This pseudocode allows for observing the behavior of k as C_{max} is incremented. The algorithm starts by defining the input data, which includes the same factors as in Algorithm 1: k_i^h, k_j^l , and v . Then, defining the output data includes a list of costs and a list of termination times (C_{max}). Define v (line 1). The $model_{lb}$ is created (considering Equations (2)–(13) of PMSPOHO model and input data) (line 2), and a $Result_{lb}$ from the minimization of $C_{max_{lb}}$ is obtained (line 3), extracting this result (line 4). A $model_{ub}$ is created (considering Equations (1), (3)–(13) of PMSPOHO model and input data) (line 5), and a $Result_{ub}$ from the minimization of k is obtained (line 6), extracting $C_{max_{ub}}$ for this result (line 7).

Algorithm 2 Progressive cost–time evaluator

Input: $\mathcal{M}, \mathcal{V}, \mathcal{T}, p_i^h, p_j^l, k_i^h, k_j^l, h, P$, and v ;
Output: $\text{ListPair}(C_{max}, k)$: List of cost respect to C_{max} .

```

1: Define  $v$ 
2:  $model_{lb} \leftarrow \text{Createmodel}(\text{Equations}(2)–(13), \text{Input})$ 
3:  $Result_{lb} \leftarrow \text{Solve}(model_{lb}, \text{Minimize})$ 
4:  $C_{max_{lb}} \leftarrow \text{ExtractCmax}(Result_{lb})$ 
5:  $model_{ub} \leftarrow \text{Createmodel}(\text{Equations}(1), (3)–(13), \text{Input})$ 
6:  $Result_{ub} \leftarrow \text{Solve}(model_{ub}, \text{Minimize})$ 
7:  $C_{max_{ub}} \leftarrow \text{ExtractCmax}(Result_{ub})$ 
8: for  $C_{max_{ref}}$  in  $C_{max_{lb}}$  to  $C_{max_{ub}}$  do
9:    $model_{ref} \leftarrow \text{Createmodel}(\text{Equations}(1), (3)–(8), (10)–(13), \text{Input})$ 
10:   $Result_{ref} \leftarrow \text{Solve}(model_{ref}, \text{Minimize})$ 
11:   $Cost \leftarrow \text{ExtractCost}(Result_{ref})$ 
12:   $\text{ListPair}(C_{max}, k) \leftarrow \text{AddToList}(C_{max_{ref}}, Cost)$ 
13: end for
14: return  $C_{max}, k$ 

```

A progressive evaluation is carried out that iterates $C_{max_{ref}}$ by 1 from $C_{max_{lb}}$ to $C_{max_{ub}}$ (line 8). For each value, a $model_{ref}$ is created (considering Equations (1), (3)–(8), and (10)–(13) of PMSPOHO model and input data) (line 9). $Result_{ref}$ is obtained by minimizing k for each iteration (line 10); k is extracted and added to a list ($Cost$) for each corresponding $C_{max_{ref}}$ (lines 11–12). This algorithm returns the list of pairs $(C_{max_{ref}}, k)$ (lines 14),

providing a detailed evaluation of how k varies about changes in $C_{max_{ref}}$. In the algorithm, $C_{max_{ref}}$ serves as an intermediate reference value to iteratively evaluate the trade-off between minimizing C_{max} and k . This iterative evaluation provides insights into the cost–time trade-offs in the model.

4. Computational Experiments and Results Analysis

This section presents the computational experiments to evaluate the performance of the proposed PMSPOHO model and algorithms. All experiments were conducted on a computer with a Windows 11 operating system, 12th Gen Intel(R) Core(TM) i5-12450H, 2.50 GHz processor, and 8 GB RAM. The Python 3.11 programming language was used for the development. The Gurobi 10.0.1 commercial optimizer and CBC free optimizer were used with a search time limit of 3600 s and a GAP of 0.01.

Based on actual data provided in the studies by Meza [2], Vásquez [3], and Anchiraico and Rojas [4], which include the numbers of loading and hauling equipment, loading times, hauling times, time horizons, and production targets, 34 experimental instances were generated. These instances are designed to simulate real-world scenarios, enabling a robust evaluation of our proposed methodologies under various conditions. This approach ensures that our experimental setup is based on realistic operational parameters, enhancing the applicability and relevance of our findings. It is important to note that a direct comparison with the authors' studies above was not conducted, as their research methodologies differ significantly from our proposed approach.

4.1. Tuning the Values for the Optimizer Parameters

Gurobi Optimizer version 10.0.1 features an extensive range of parameters designed to fine-tune the optimizer performance. Given the vast array of options among possible configurations, identifying the optimal combination of parameters that enhance the overall model performance poses a significant challenge. The TuningAPI from Gurobi offers a valuable tool for exploring various parameter settings that can improve model performance in specific cases [56].

Considering the procedure indicated by Cuellar-Usaquén et al. [57], a sample of seven instances was created to identify the appropriate parameter settings for our model using TuningAPI. This sample includes those instances where the computation time obtained by the PMSPOHO model using the default parameters (PMSPOHO-D) exceeds the meantime of its respective class by two standard deviations.

Table 3 shows the results of Gurobi parameter tuning across seven different problem instances. Each table row represents an instance, with columns indicating the specific Gurobi parameters adjusted for that instance. The parameters include Symmetry, DegenMove, Heuristics, GomoryPasses, PreDepRow, MIPFocus, Method, CutPasses, and BranchDir.

Table 3. Parameters configurations found for the mathematical model.

Summary				Value Parameters									Tc-D	Tc-T	Variation
Inst	v	m	P	Symmetry	DegenMoves	Heuristics	GomoryPasses	PreDepRow	MIPFocus	Method	CutPasses	BranchDir			
1	3	2	20	2									0.8	0.79	1.25%
2	6	2	20		2								1.22	1.15	5.74%
3	8	4	81			0	1	1					69.59	55.03	20.92%
4	12	4	81						2				46.54	44.4	4.60%
5	15	4	81							0			32.53	31.64	2.74%
6	8	4	90				1						139.22	120.68	13.32%
7	20	4	100								1	1	81.62	79.9	2.11%

The third instance stands out significantly because it shows the highest variation (20.92%) between the default computational time (Tc-D) and the tuned computational time (Tc-T), indicating a significant improvement in performance with the adjusted parameters. In this instance, the parameters Heuristics, GomoryPasses, and PreDepRow were adjusted, resulting in a substantial reduction in computation time from 69.59 to 55.03 time units. This adjustment reduced the overall calculation time and optimized the use of computational resources, making the process more efficient and effective. The con-

siderable difference between Tc-D and Tc-T underscores the importance of customization and fine-tuning model parameters to achieve maximum efficiency in solving complex mathematical problems.

In this instance, the adjusted parameter Heuristics emerges as a key player in accelerating the solving process. By increasing the use of heuristic methods, which are solving strategies that employ simple and fast rules to generate satisfactory solutions quickly, the model explored and evaluated potential solutions more efficiently. This approach significantly reduced the time required to reach a feasible solution, demonstrating its practicality and effectiveness. Applying heuristics is a valuable tool, particularly in scenarios where speed is critical and some precision can be sacrificed for improved resolution speed.

GomoryPasses, another adjusted parameter, controls the amount of Gomory cut passes. These passes are optimization techniques that strengthen the model linear relaxation, leading to faster convergence. Adjusting the amount of these passes can significantly impact the solution efficiency, balancing the need to improve the solution with the additional time required for each cut. PreDepRow, which handles the preprocessing of dependent rows, also fundamentally improved efficiency by reducing problem size and enhancing numerical stability. Effective preprocessing can simplify the model by eliminating redundancies and reducing overall complexity. In the third instance, adjusting PreDepRow significantly reduced computational time, noting the importance of proper model management. These adjustments underscore the importance of tuning parameters to enhance the efficiency solver, enabling faster and more accurate results.

4.2. Benchmarking Results for Optimizers

Table 4 presents the benchmarking results of optimizers across 34 instances. The instances were organized considering the production target, as an increase in computational time (Tc) was noted as this target increased. The results were compared between the default model (PMSPHOH-D), the tuned model (PMSPHOH-T), and the model with the free optimizer (PMSPHOH-CBC). The critical parameters for comparison were Tc and GAP, which measure the complexity model and the efficiency solver. In this study, the GAP value is provided directly by the Gurobi optimizer as part of its solution output.

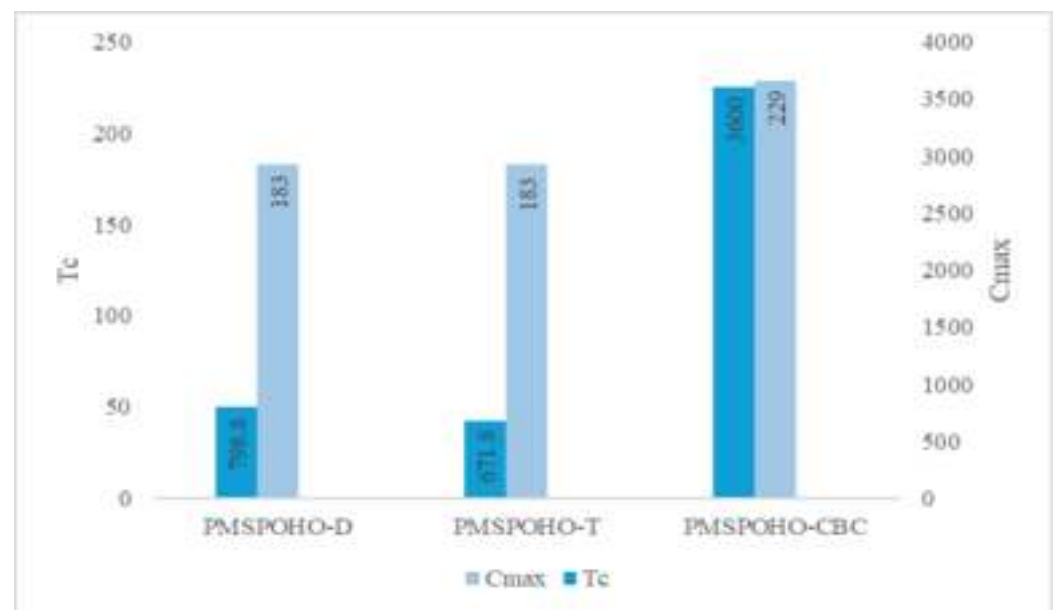
The tuned model (PMSPHOH-T) consistently handles lower computational times than the other models. This indicates that parameter tuning has significantly improved the solver efficiency. The GAP, which is the difference between the best-known solution and the best lower bound found by the solver, is 0.00% for both PMSPHOH-D and PMSPHOH-T, indicating that the solutions found are optimal. However, PMSPHOH-CBC shows a higher GAP in several instances, reflecting lower precision and highlighting the limitations of a commercial optimizer.

PMSPHOH-T stands out with the best results in terms of Tc, solving instances faster while maintaining solution precision. This starkly contrasts with the model with the free optimizer (PMSPHOH-CBC), which shows a higher GAP in many cases, thereby highlighting its limitations. It also demonstrates that the problem's complexity is NP-Hard, meaning that as instances increase, optimal results could be obtained after significantly longer times, making a free optimizer an inefficient option for field applications.

Figure 3 compares the computational time (Tc) and $C_{\max}$ for the PMSPHOH-D, PMSPHOH-T, and PMSPHOH-CBC optimizers for instance 34. PMSPHOH-D and PMSPHOH-T achieve the same $C_{\max}$ of 183, but PMSPHOH-T is more efficient in terms of computational time, with 671.8 compared to 798.8 for PMSPHOH-D. On the other hand, PMSPHOH-CBC shows the worst performance, with a $C_{\max}$ of 229 and a computational time of 3600. This indicates that PMSPHOH-T is the most efficient among the three evaluated optimizers, achieving a good balance between computational time and $C_{\max}$.

Table 4. Benchmarking results for optimizers per instances.

Instance	P	v	m	PMSPOHO-D			PMSPOHO-T			PMSPOHO-CBC		
				C_{max}	Tc	GAP	C_{max}	Tc	GAP	C_{max}	Tc	GAP
1	20	20	2	29	14.52	0.00%	29	9.73	0.00%	30	3600	4.58%
2	20	16	2	29	9.51	0.00%	29	5.92	0.00%	30	3600	4.64%
3	20	10	2	29	3.82	0.00%	29	2.91	0.00%	31	3600	5.21%
4	20	8	2	29	2.51	0.00%	29	1.85	0.00%	30	3600	4.82%
5	20	6	2	31	1.6	0.00%	31	1.58	0.00%	32	3600	3.77%
6	20	5	2	34	1.41	0.00%	34	1.03	0.00%	35	3600	2.36%
7	20	4	2	40	1.13	0.00%	40	1.01	0.00%	41	3600	2.56%
8	20	3	2	51	0.9	0.00%	51	1.82	0.00%	51	1846	0.00%
9	81	20	4	55	28.25	0.00%	55	26.74	0.00%	60	3600	8.52%
10	81	16	4	55	18.34	0.00%	55	18.61	0.00%	60	3600	8.20%
11	81	10	4	64	49.23	0.00%	64	46.08	0.00%	68	3600	6.45%
12	81	8	4	78	69.59	0.00%	78	50.86	0.00%	82	3600	5.50%
13	81	6	4	100	66.54	0.00%	100	63.01	0.00%	106	3600	5.72%
14	81	4	4	147	76.29	0.00%	147	59.78	0.00%	152	3600	3.27%
15	81	20	6	38	42.12	0.00%	38	42.26	0.00%	42	3600	10.25%
16	81	16	6	42	318.67	0.00%	42	130.11	0.00%	46	3600	9.86%
17	81	10	6	63	94.3	0.00%	63	76.82	0.00%	70	3600	10.52%
18	81	8	6	77	100.46	0.00%	77	100.13	0.00%	85	3600	10.73%
19	81	6	6	98	145.77	0.00%	98	90.32	0.00%	110	3600	12.72%
20	90	5	3	130	42.98	0.00%	130	26.41	0.00%	163	3600	25.32%
21	90	4	3	161	83.77	0.00%	161	40.36	0.00%	202	3600	25.47%
22	90	3	3	212	44.52	0.00%	212	30.78	0.00%	259	3600	22.23%
23	200	19	4	125	1784.21	0.00%	125	1627.21	0.00%	161	3600	29.03%
24	200	18	4	125	1693.64	0.00%	125	1437.64	0.00%	161	3600	28.53%
25	200	17	4	125	1634.63	0.00%	125	1463.63	0.00%	161	3600	28.93%
26	200	16	4	125	1640.24	0.00%	125	1400.24	0.00%	161	3600	28.73%
27	200	15	4	125	1584.04	0.00%	125	1437.04	0.00%	164	3600	31.45%
28	200	14	4	125	1554.43	0.00%	125	1291.43	0.00%	159	3600	27.32%
29	200	13	4	125	1532.23	0.00%	125	1349.23	0.00%	161	3600	29.05%
30	200	12	4	151	1524.64	0.00%	151	1372.64	0.00%	194	3600	28.63%
31	200	11	4	151	1279.45	0.00%	151	1019.45	0.00%	189	3600	25.43%
32	200	10	4	151	1017.6	0.00%	151	746.6	0.00%	186	3600	23.23%
33	200	9	4	166	865.6	0.00%	166	615.6	0.00%	203	3600	22.01%
34	200	8	4	183	798.8	0.00%	183	671.8	0.00%	229	3600	24.93%

**Figure 3.** Comparison of computational time and C_{max} for instance 34.

4.3. Tactical Approach Insights

Table 5 shows the efficiency of the applied algorithms. Algorithm 1 calculates the minimum number of hauling equipment, resulting in eight equipment. On the other hand, Algorithm 2 performs a k - C_{max} evaluation, allowing for a more appropriate analysis between these two variables. This improves tactical decision-making by showing how varying C_{max} impacts k and vice-versa. Minimizing C_{max} results in significantly reduced values (between 33 and 49), although an increase in k enables planners to balance temporal efficiency and operational costs. Algorithm 2 provides a more comprehensive and balanced

perspective for tactical decision-making and is supported by Algorithm 1, which defines the number of resources needed to meet the production target.

Table 5. Results of optimizing the proposed algorithms.

v	m	P	Minimize k			Minimize C_{max}		
			C_{maxub}	k	T_c	C_{maxlb}	k	T_c
19	4	200	51	2350	151.46	33	6998	54.97
18	4	200	51	2350	219.68	33	6648	46.31
17	4	200	51	2350	117.08	33	6298	38.57
16	4	200	51	2350	99.44	33	5948	33.81
15	4	200	51	2350	125.16	33	5598	31.16
14	4	200	51	2350	162.80	33	5248	26.78
13	4	200	51	2350	75.34	34	4898	24.63
12	4	200	51	2350	58.55	35	4548	20.73
11	4	200	51	2350	111.22	37	4198	21.40
10	4	200	51	2350	103.30	40	3848	32.76
9	4	200	50	2350	44.62	44	3498	19.40
8	4	200	51	2350	76.94	49	3148	15.93

As completion times shorten, costs rise disproportionately, indicating a trade-off between time and cost efficiency in ore hauling operations. The steeper curves at lower completion times imply significant cost increases for faster operations, while the flatter slopes at higher times suggest a minimum cost threshold. The broader range of completion times with more hauling equipment means that, although additional equipment can speed up operations, it also incurs higher costs, highlighting the importance of balancing budgetary limitations with operational urgency in scheduling mining operations.

Figure 4 shows the relationship between costs and completion time for different numbers of hauling equipment. There is a marked cost decrease with only a slight increase in completion time. A tactical approach to determining the minimum amount of hauling equipment needed to meet production targets should consider not only cost and completion time but also the slope of each line, which supports the application of the algorithms proposed in this paper. For instance, moving from nine to eight hauling units marginally increases completion time while significantly reducing cost. This suggests that operating with eight hauling units could be an effective tactical decision if the additional time does not compromise other priorities of the ore hauling operation.

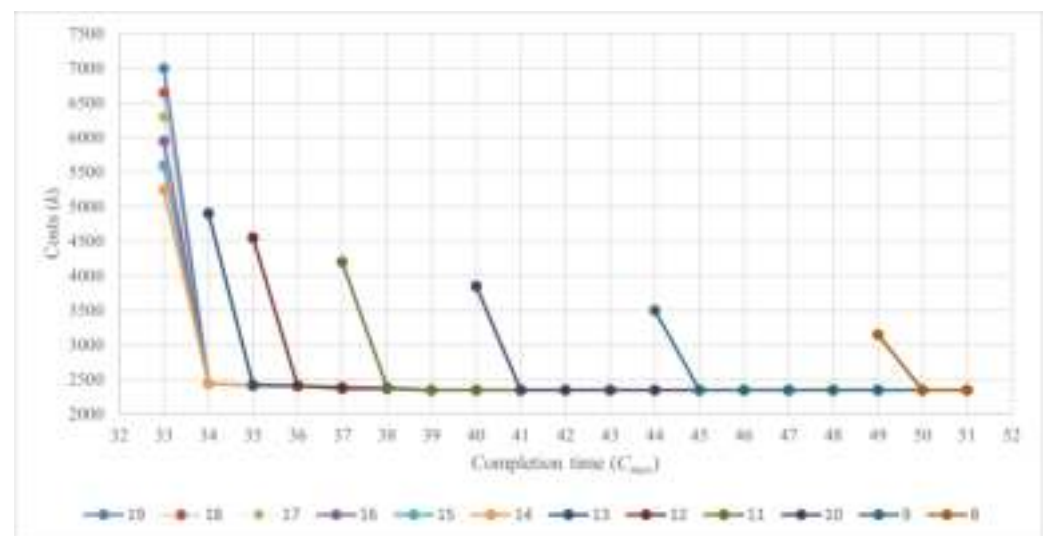


Figure 4. Costs k vs. completion time C_{max} for the instances proposed in Table 5.

Decision-makers must weigh the cost increase associated with using more equipment against reducing hauling time. Beyond a certain point, adding more hauling units diminishes the benefit of lowering completion time but incurs substantially higher costs. Pinpointing the moment when the additional cost no longer justifies the decrease in hauling time is critical for optimizing operations and managing resources efficiently.

4.4. Operational Approach Insights

Figure 5 shows the schedule, for instance 34, which considers eight hauling and four loading equipment. This visualizes the ore loading program for a mining operation during the first 50 trips of the production target. It is interesting to see how the allocation is organized to ensure that the minimum necessary amount of hauling equipment is kept in constant motion, indicating a careful assessment of loading times and intervals between tasks. This strategy ensures that no loading equipment remains idle and that the hauling equipment is effectively distributed to meet the production target.

By minimizing the amount of hauling equipment, the operation can become more cost-effective and improve logistics by reducing congestion and potential bottlenecks in the ore hauling operation. This balanced approach between efficiency and effectiveness in scheduling is crucial for a successful operation that adapts to the dynamic and often unpredictable demands of mining.

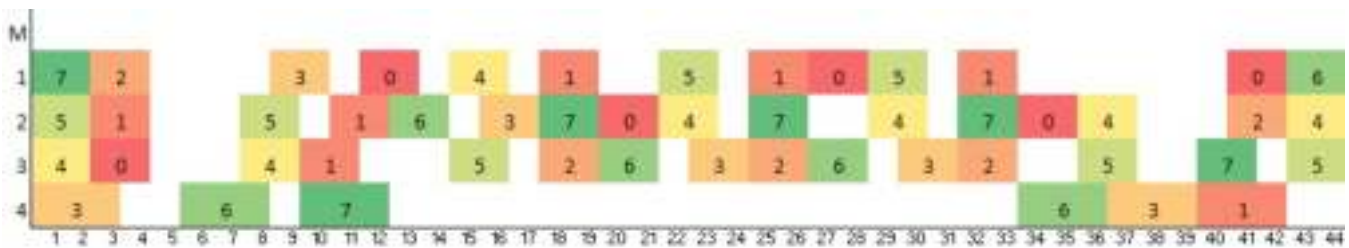


Figure 5. Schedule for instance 34 considering minimize completion time C_{max} .

Table 6 provides a detailed distribution of loading and hauling equipment utilization in a mining operation, highlighting the allocation and usage of these resources in production. It is observed that loading equipment two and three are the most utilized, with utilization percentages of 32% and 30%, respectively, indicating high demand and frequent use. In contrast, loading vehicle four is the least utilized, operating at only 12% utilization. This suggests that resources are not being fully optimized, particularly considering that loading vehicle four might be the most costly to operate, as less frequently used equipment typically requires regular maintenance and occupies valuable space that could be better utilized.

The under-utilization of the costly loading equipment four raises significant cost management and operational efficiency considerations. The high cost of this equipment may be due to specialized features that necessitate more sophisticated maintenance or a higher initial investment. However, its sub-optimal utilization implies that this investment needs to yield the expected returns, which is critical in operations aiming to maximize efficiency and profitability. This situation underscores the urgent need to reassess task distribution and resource allocation to ensure that all loading equipment are utilized in a manner that justifies their cost, thereby optimizing the entire operation and reducing potential resource wastage.

The results demonstrate that the proposed model is efficient in terms of computational time and that the proposed algorithms are useful tactical tools for more effective decision-making regarding the allocation of hauling equipment necessary to achieve the targets set in ore transport operations. However, it is essential to acknowledge that analyzing more significant instances as a mathematical model could complicate the time required to obtain results, even with the optimizer parameters finely tuned. Therefore, it will be essential to extend the solution search periods and develop approximate methods to reduce these times and attain practical and executable solutions.

Table 6. Loading and hauling equipment utilization.

Hauling Equipment	Loading Equipment				Total	Utilization
	1	2	3	4		
0	3	2	1	0	6	12%
1	3	2	1	1	7	14%
2	1	1	3	0	5	10%
3	1	1	2	2	6	12%
4	1	4	2	0	7	14%
5	2	2	3	0	7	14%
6	1	1	2	2	6	12%
7	1	3	1	1	6	12%
Total	13	16	15	6	50	
Utilization	26%	32%	30%	12%		

5. Conclusions

The proposed model demonstrates significant efficiency in terms of computational time, effectively balancing cost and completion time in ore hauling operations. Applying the two algorithms provided tactical tools that enhance decision-making regarding the allocation of hauling equipment. The model efficiency is evident in the computational results, showing that parameter tuning can significantly improve solver performance. The tuned model consistently handled lower computational times than default settings, highlighting the importance of careful parameter adjustments. Furthermore, incorporating dynamic release times and different speeds of hauling equipment in the model adds a layer of realism, ensuring that the scheduling reflects actual operational conditions. This aspect further optimizes the allocation process, making the model computationally robust.

Despite the results, the model has limitations. A primary limitation is its scalability. As the size of the instances increases, the time required to obtain results can become prohibitive, even with optimized parameters. This issue is inherent in mathematical models dealing with complex, NP-hard problems. Additionally, the reliance of the model on exact optimization methods may only sometimes be practical for real-time decision-making in dynamic environments where quick responses are essential.

Future work should focus on extending solution search periods and developing approximate methods to address these limitations. These methods, including heuristic or metaheuristic approaches, can reduce computational times and provide near-optimal practical and executable solutions in real-world scenarios. Additionally, integrating machine learning techniques to predict and dynamically adjust parameters could enhance the model's performance and adaptability [58].

Author Contributions: Conceptualization, L.T.-T., C.A., and D.A.-M.; methodology, L.T.-T. and C.A.; software, L.T.-T. and C.G.; formal analysis, L.T.-T. and A.P.; investigation, L.T.-T. and A.P.; data curation, L.T.-T.; writing—original draft preparation, L.T.-T.; writing—review and editing, L.T.-T., C.A., and D.A.-M.; visualization, L.T.-T.; supervision, C.A. and D.A.-M.; project administration, L.T.-T. All authors have read and agreed to the published version of the manuscript.

Funding: The APC was funded by University of Los Andes.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on: <https://github.com/LuisTarazonaTorres/Parallel-Machine-Scheduling-problem---Hualing-Equipment.git> (accessed on 29 July 2024).

Acknowledgments: The authors greatly acknowledge all the participants who contributed to the conduct of this study.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Rout, M.; Parida, C.K. Optimization of Blasting Parameters in Opencast Mines. Ph.D. Thesis, National Institute of Technology Rourkela, Odisha, India, 2007.
2. Meza Castro, J.E. Desarrollo de un Modelo para la Aplicación de Simulación a un Sistema de Carguío y Acarreo de Desmonte en una Operación Minera a Tajo Abierto. Bachelor's Thesis, Pontifical Catholic University of Peru, San Miguel, Peru, 2011.
3. Vasquez, O.C.; Sepúlveda, J.M.; Córdova, F. Modeling and optimization of vehicle operations in underground copper mining. *Rev. Rev. Esc. De Minas* **2011**, *64*, 365–371. [\[CrossRef\]](#)
4. Anchiraico, A.; Rojas, K. Optimización del Sistema de Acarreo y Transporte en Labores de Preparación de las Zonas de Profundización Mediante la Metodología Six Sigma Operada por la E.C.M. Zicsa en la Unidad Minera Inmaculada. Bachelor's Thesis, Universidad Peruana de Ciencias Aplicadas, Santiago de Surco, Peru, 2023.
5. Soofastaei, A.; Karimpour, E.; Knights, P.; Kizil, M. *Energy-Efficient Loading and Hauling Operations*; Springer: Cham, Switzerland, 2018; pp. 121–146. [\[CrossRef\]](#)
6. Choi, Y.; Baek, J.; Park, S. Review of GIS-based applications for mining: Planning, operation, and environmental management. *Appl. Sci.* **2020**, *10*, 2266. [\[CrossRef\]](#)
7. Owolabi, A. Loading and haulage equipment selection for optimum production in a granite quarry. *Int. J. Min. Sci.* **2019**, *5*, 35–40.
8. Marte Collado, J.M. Programación de la Producción en Máquinas Paralelas Sujeto a Adelantos, Retrasos y Fechas límite. Master's Thesis, Universitat Politècnica de València, Valencia, Spain, 2017.
9. Salazar-Hornig, E.; Soto-Gavilán, G. Minimización del makespan para el problema de máquinas paralelas no relacionadas con tiempos de setup dependientes de la secuencia mediante un algoritmo híbrido VNS/ACO. *Rev. Ing. Univ. De Medellín* **2021**, *20*, 171–184. [\[CrossRef\]](#)
10. Hong, H.C.; Lin, B.M. Parallel dedicated machine scheduling with conflict graphs. *Comput. Ind. Eng.* **2018**, *124*, 316–321. [\[CrossRef\]](#)
11. Koryagin, M.; Voronov, A. Improving the organization of the shovel-truck systems in open-pit coal mines. *Transp. Probl.* **2017**, *12*, 113–122. [\[CrossRef\]](#)
12. Olishevych, M.; Kovalyshyn, S.; Magats, M.; Shevchuk, V.; Sukach, O. The optimization of trucks fleet schedule in view of their interaction and restrictions of the European agreement of work of crews. *Transp. Probl.* **2020**, *15*, 157–170. [\[CrossRef\]](#)
13. Wu, L.; Wang, S. Exact and heuristic methods to solve the parallel machine scheduling problem with multi-processor tasks. *Int. J. Prod. Econ.* **2018**, *201*, 26–40. [\[CrossRef\]](#)
14. Elidrissi, A.; Benmansour, R.; Benbrahim, M.; Duvivier, D. Mathematical formulations for the parallel machine scheduling problem with a single server. *Int. J. Prod. Res.* **2021**, *59*, 6166–6184. [\[CrossRef\]](#)
15. Gan, H.S.; Wirth, A.; Abdekhodae, A. A branch-and-price algorithm for the general case of scheduling parallel machines with a single server. *Comput. Oper. Res.* **2012**, *39*, 2242–2247. [\[CrossRef\]](#)
16. Kim, M.Y.; Lee, Y.H. MIP models and hybrid algorithm for minimizing the makespan of parallel machines scheduling problem with a single server. *Comput. Oper. Res.* **2012**, *39*, 2457–2468. [\[CrossRef\]](#)
17. Hasani, K.; Kravchenko, S.A.; Werner, F. Minimising interference for scheduling two parallel machines with a single server. *Int. J. Prod. Res.* **2014**, *52*, 7148–7158. [\[CrossRef\]](#)
18. Kosch, S.; Beck, J.C. A new mip model for parallel-batch scheduling with non-identical job sizes. In Proceedings of the Integration of AI and OR Techniques in Constraint Programming: 11th International Conference (CPAIOR 2014), Cork, Ireland, 19–23 May 2014; Proceedings 11; Springer: Berlin/Heidelberg, Germany, 2014; pp. 55–70.
19. Sharma, A.; Zhao, F.; Sutherland, J.W. Econological scheduling of a manufacturing enterprise operating under a time-of-use electricity tariff. *J. Clean. Prod.* **2015**, *108*, 256–270. [\[CrossRef\]](#)
20. Bajany, D.; Xia, X.; Zhang, L. A MILP Model for Truck-shovel scheduling to minimize fuel consumption. *Energy Procedia* **2017**, *105*, 2739–2745. [\[CrossRef\]](#)
21. Che, A.; Zhang, S.; Wu, X. Energy-conscious unrelated parallel machine scheduling under time-of-use electricity tariffs. *J. Clean. Prod.* **2017**, *156*, 688–697. [\[CrossRef\]](#)
22. Cheng, J.; Chu, F.; Zhou, M. An improved model for parallel machine scheduling under time-of-use electricity price. *IEEE Trans. Autom. Sci. Eng.* **2017**, *15*, 896–899. [\[CrossRef\]](#)
23. Elidrissi, A.; Benmansour, R.; Benbrahim, M.; Duvivier, D. MIP formulations for identical parallel machine scheduling problem with single server. In Proceedings of the 2018 4th International Conference on Optimization and Applications (ICOA), Mohammedia, Morocco, 26–27 April 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 1–6.
24. Kaabi, J.; Harrath, Y. Scheduling on uniform parallel machines with periodic unavailability constraints. *Int. J. Prod. Res.* **2019**, *57*, 216–227. [\[CrossRef\]](#)
25. Quiroz Silva, E.E. Programación por Objetivos para el Dimensionamiento y la Asignación de Una Flota de Camiones en una Empresa Minera. *Interfases* **2021**, 87–112. [\[CrossRef\]](#)
26. Hsu, C.L.; Liao, J.R. Two parallel-machine scheduling problems with function constraint. *Discret. Dyn. Nat. Soc.* **2020**, *2020*, 1–6. [\[CrossRef\]](#)

27. Tapia, E.; Salazar Araya, A.; Saavedra, N.; Nehring, M.; Mora, J. An analysis of full truck versus full bucket strategies in open pit mining loading and hauling operations. *Int. J. Min. Reclam. Environ.* **2021**, *35*, 1–11. [\[CrossRef\]](#)
28. Tigreros Cardenas, I.D. Propuesta Para la Programación de Producción en Máquinas Paralelas con Tiempos de Alistamiento en una Empresa del Sector Agroindustrial. 2021. Available online: <https://repositorio.uceva.edu.co/bitstream/handle/20.500.129.93/2744/T00031983.pdf?sequence=1&isAllowed=y> (accessed on 29 July 2024).
29. Yunusoglu, P.; Topaloglu Yildiz, S. Constraint programming approach for multi-resource-constrained unrelated parallel machine scheduling problem with sequence-dependent setup times. *Int. J. Prod. Res.* **2022**, *60*, 2212–2229. [\[CrossRef\]](#)
30. Ma, X.; Fu, Y.; Gao, K.; Zhu, L.; Sadollah, A. A multi-objective scheduling and routing problem for home health care services via brain storm optimization. *Complex Syst. Model. Simul.* **2023**, *3*, 32–46. [\[CrossRef\]](#)
31. Feng, L.; Chen, G.; Zhou, S.; Zhou, X.; Jin, M. An Energy-Efficient Unrelated Parallel Machine Scheduling Problem with Batch Processing and Time-of-Use Electricity Prices. *Mathematics* **2024**, *12*, 376. [\[CrossRef\]](#)
32. Hu, B.; Xiong, Z.; Sun, A.; Yuan, Y. Scheduling of Container Transportation Vehicles in Surface Coal Mines Based on the GA–GWO Hybrid Algorithm. *Appl. Sci.* **2024**, *14*, 3986. [\[CrossRef\]](#)
33. Eivazy, H.; Askari-Nasab, H. A mixed integer linear programming model for short-term open pit mine production scheduling. *Min. Technol.* **2012**, *121*, 97–108. [\[CrossRef\]](#)
34. Anjomshoa, H.; Albrecht, A.; Lee, D.; Pudney, P. Efficient vehicle haulage in underground mines. *Min. Technol.* **2012**, *121*, 83–90. [\[CrossRef\]](#)
35. Gligoric, Z. Optimization of Haulage System in An Underground Small Scale Mine Using Fuzzy Sets. Systems Design. p. 63. Available online: https://www.researchgate.net/publication/287397565_Optimization_of_haulage_system_in_an_underground_small_scale_mine_using_fuzzy_sets (accessed on 29 July 2024).
36. Gonen, A.; Malli, T.; Kose, H. Selection of ore transport system for a metalliferous underground mine/Dobór systemu transportu rud w kopalni podziemnej rud metalu. *Arch. Min. Sci.* **2012**, *57*, 779–785. [\[CrossRef\]](#)
37. Eivazy, H.; Askari-Nasab, H. A hierarchical open-pit mine production scheduling optimisation model. *Int. J. Min. Miner. Eng.* **2012**, *4*, 89–115. [\[CrossRef\]](#)
38. Tom Socarras, J.N. Árbol de Decisión para la Gestión y Control de Costo de Transporte Interior Mina. Bachelor’s Thesis, University of Chile, Santiago, Chile, 2014.
39. Uribe Neira, P.A. Análisis de la Funcionalidad y Desempeño de la Operación Semiautónoma en Equipos de Carga; Acarreo y Descarga en Minería Subterránea Load Haul-Dump Vehicle (LHD). Bachelor’s Thesis, University of Chile, Santiago, Chile, 2014.
40. Więcek, D.; Burduk, A.; Kuric, I. The use of ANN in improving efficiency and ensuring the stability of the copper ore mining process. *Acta Montan. Slovaca* **2019**, *24*, 1–14.
41. Wiest Goyeneche, J.R. Modelo de Programación de Operaciones en una Mina a Cielo Abierto: Aplicación en Organización Corona. Master’s Thesis, Universidad de La Sabana, Chía, Colombia, 2019.
42. Solomon, L.; Ortiz, A. Conceptual model of open pit ore loading and transport systems. *Univ. Cienc. Y Tecnol.* **2020**, *24*, 41–50.
43. Gaspar, S.; Jhasmin, C. Mejora de la Productividad en Equipos de Acarreo y Transporte de Mineral y Desmonte en la Veta Gavia–Nivel 100, Unidad Minera Huarón. Bachelor’s Thesis, Universidad Continental, Los Olivos, Peru, 2020.
44. Li, N.; Feng, S.; Ye, H.; Wang, Q.; Jia, M.; Wang, L.; Zhao, S.; Chen, D. Dispatch Optimization Model for Haulage Equipment between Stopes Based on Mine Short-Term Resource Planning. *Metals* **2021**, *11*, 1848. [\[CrossRef\]](#)
45. Elijah, K.; Joseph, G.; Samuel, M.; Mauti, D. Optimisation of shovel-truck haulage system in an open pit using queuing approach. *Arab. J. Geosci.* **2021**, *14*, 973. [\[CrossRef\]](#)
46. Huisa Supho, A. Optimización de Equipos de Carguío y Acarreo en el Tramo Botaderos-Trituradora Thyssen Krupp, para el Incremento de Producción en una Empresa Cementera. Bachelor’s Thesis Universidad Tecnológica Del Perú, Lima, Peru, 2021.
47. Choi, Y.; Nguyen, H.; Bui, X.N.; Nguyen-Thoi, T. Optimization of haulage-truck system performance for ore production in open-pit mines using big data and machine learning-based methods. *Resour. Policy* **2022**, *75*, 102522. [\[CrossRef\]](#)
48. Li, N.; Feng, S.; Lei, T.; Ye, H.; Wang, Q.; Wang, L.; Jia, M. Rescheduling Plan Optimization of Underground Mine Haulage Equipment Based on Random Breakdown Simulation. *Sustainability* **2022**, *14*, 3448. [\[CrossRef\]](#)
49. Abolghasemian, M.; Kanafi, A.G.; Daneshmand-Mehr, M. Simulation-based multiobjective optimization of open-pit mine haulage system: A modified-NBI method and meta modeling approach. *Complexity* **2022**, *2022*, 3540736. [\[CrossRef\]](#)
50. Shamsi, M.; Pourrahimian, Y.; Rahmanpour, M. Optimisation of open-pit mine production scheduling considering optimum transportation system between truck haulage and semi-mobile in-pit crushing and conveying. *Int. J. Min. Reclam. Environ.* **2022**, *36*, 142–158. [\[CrossRef\]](#)
51. Parichehreh, M.; Gholizadeh, H.; Fathollahi-Fard, A.; Wong, K. An energy-efficient unrelated parallel machine scheduling problem with learning effect of operators and deterioration of jobs. *Int. J. Environ. Sci. Technol.* **2024**, 1–26. [\[CrossRef\]](#)
52. Graham, R.L.; Lawler, E.L.; Lenstra, J.K.; Kan, A.R. Optimization and approximation in deterministic sequencing and scheduling: a survey. In *Annals of Discrete Mathematics*; Elsevier: Amsterdam, The Netherlands, 1979; Volume 5, pp. 287–326.
53. Pinedo, M.; Hadavi, K. Scheduling: Theory, algorithms and systems development. In *Proceedings of the Operations Research Proceedings 1991: Papers of the 20th Annual Meeting/Vorträge der 20. Jahrestagung*; Springer: Berlin/Heidelberg, Germany, 1992; pp. 35–42.
54. Yu, K.; Yang, J. MILP model and a rolling horizon algorithm for crane scheduling in a hybrid storage container terminal. *Math. Probl. Eng.* **2019**, *2019*, 4739376. [\[CrossRef\]](#)

55. Tarazona-Torres, L.E.; Amaya-Guio, C.A.; Álvarez-Martínez, D. Asynchronous Team for Flow Shop Scheduling Problem: A Case Study. In Proceedings of the 4th South American International Conference on Industrial Engineering and Operations Management, Lima, Peru, 9–11 May 2023.
56. Gurobi Optimization, LLC. *Gurobi Optimizer Parameters Version 10.0.2*; Gurobi Optimization, LLC.: Beaverton, OR, USA, 2024.
57. Cuellar-Usaquén, D.; Palacio, A.; Ospina, E.; Botero, M.; Álvarez-Martínez, D. Modeling and solving the endpoint cutting problem. *Int. Trans. Oper. Res.* **2023**, *30*, 800–830. [[CrossRef](#)]
58. Hou, Y.; Wang, H.; Huang, X. A Q-learning-based multi-objective evolutionary algorithm for integrated green production and distribution scheduling problems. *Eng. Appl. Artif. Intell.* **2024**, *127*, 107434. [[CrossRef](#)]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.