Received: 2 September 2024 | Revised: 11 November 2024 | Accepted: 21 November 2024 | Published online: 31 December 2024

RESEARCH ARTICLE

Journal of Computational and Cognitive Engineering
2025, Vol. 4(3) 387–396
DOI: 10.47852/bonviewJCCE42024233

BON VIEW PUBLISHING

Whale Optimization Algorithm for Feature Selection Enhances Classification in Malware Datasets

Mariam Al Ghamri[1,*], **Dyala Ibrahim**[1], **Rami Sihwail**[1] **and Mohammad Shehab**[1]

[1] *College of Computer Sciences and Informatics, Amman Arab University, Jordan*

Abstract: Malicious programs are increasing abnormally, affecting our everyday lives. Modern sophisticated and agile malware programs are not always detected by traditional malware detection methods that use signature-based techniques. As a result, researchers use behavior-based techniques to analyze malware behaviors (features). However, malware features derived from behavioral analysis commonly suffer from high dimensionality. Accordingly, this work applies the Whale Optimization Algorithm (WOA) to find the optimal subset of features in the CIC-MalMem-2022 dataset. Feature selection contributes significantly to reducing high-dimensionality issues and improving malware detection performance. WOA is employed to enhance the efficiency of the selection process for the optimal features and determine the most advantageous set of features by omitting redundant and irrelevant features. In addition, we apply the K-nearest neighbor algorithm (KNN) to detect malware. Using WOA and KNN, this study improves the detection efficiency of CIC-MalMem-2022. According to the findings, the proposed method outperforms existing malware detection systems, including detection fitness value, accuracy, consuming time, and the number of selected features.

Keywords: malware, K-nearest neighbor algorithm (KNN), feature selection, Whale Optimization Algorithm (WOA), classification

1. Introduction

Malware detection systems refer to software applications that are specifically engineered to penetrate or impair a computer system with a consent from the user, hence posing a significant risk to the overall security of the system. Emerging viral strains, including polymorphic variants, employ sophisticated obfuscation methods, rendering them far more intricate and elusive in terms of detection. In contemporary, malware detection systems, one of the predominant methods employed by malware detection software involves the utilization of techniques that rely on identifying patterns and those that monitor actions to recognize potential threats. Signatures refer to concise sequences of bytes that possess program-specific characteristics [1]. The aforementioned technique has a low error rate and possesses the capability to discern certain viruses that are live in electronic files, boot data, or data repositories [2]. However, the efficiency of the signature-based method is compromised when faced with malicious executables that have changed or are unknown. Behavioral-based recognition has a higher propensity to protect against novel and unforeseen risks. Nevertheless, the process of detecting this particular form of virus is commonly associated with significant expenses and is time-consuming [3]. In the past, malware programs were developed using simple coding techniques, which made them prone to detection, especially by sophisticated and agile security systems. In contemporary times, there has been a discernible trend among malware creators to enhance the complexity of their code, leading to a situation where even sophisticated detection methods are ineffective. Differentiating next-generation malware files from traditional malware, which is specifically engineered to operate within the kernel, poses a considerable challenge [1]. These forms of malicious software can evade security systems such as firewalls and antivirus software with relative ease. The entities exhibit a persistent presence within a system or network, proliferating across various extensions, and engaging in targeted attacks against the user. There exist multiple malware detection approaches that rely on diverse features such as signature-based, heuristic-based, behaviour-based, model checking, cloud-deployed, mobile device-centric, Internet of Things (IoT)-oriented, machine learning-driven, and deep learning-powered approaches [4]. In the initial stages, conventional techniques that rely on signature-based detection are commonly utilized for identifying malware. Although it possesses speed, it cannot detect complex malware [5].

The conventional approach of utilizing signature-based methods, such as pattern matching, proved to be inadequate in fulfilling the necessary criteria for effective malware detection [6]. Therefore, the implementation of advanced security approaches poses significant challenges for both end users and security providers [7]. Additional novel techniques are suggested, drawing upon a range of features such as heuristics and model checking. The utilization of sophisticated data mining and machine learning algorithms is integrated into these methodologies to recognize and discover harmful programs. Although signature-based detection is fast and efficient at identifying known threats, it struggles when dealing with new or modified malware variants. The main issue is that signature-based methods cannot keep up with the constantly changing nature of malware, especially with the rise of polymorphic strains and

*Corresponding author: Mariam Al Ghamri, College of Computer Sciences and Informatics, Amman Arab University, Jordan. Email: m.alghamry@aau.edu.jo

obfuscation techniques. As a result, it falls short in providing complete protection [8].

As an alternative, behavior-based detection tackles the problem by monitoring the actions of software and spotting patterns that suggest malicious activity [9]. This approach has the advantage of being able to adapt to new threats. However, it comes with its own set of challenges, particularly the need to analyze a large number of features, which can result in complex, high-dimensional problems in the feature space [10]. The abundance of features not only complicates the analysis but also imposes computational burdens, hindering the overall efficiency of the detection process [11]. Feature selection aims to optimize the performance of behavior-based detection, enhancing accuracy and efficiency in distinguishing between malicious and benign software.

Feature selection is crucial in machine learning since it serves as the most potent defense against the "curse of dimensionality" problem. Usually, the feature space consists of a variety of features that can be either valid, irrelevant, or redundant. Undoubtedly, the inclusion of unnecessary and duplicative features reduces the efficiency of the classification process. According to this, the selection of optimal features aims to address the issue of high-dimensional feature spaces, decreasing the execution time, and improving classification performance [12]. Logically, the best feature subset consists of the fewest features while achieving the highest classification performance. Optimization algorithms are well-known algorithms that have been utilized for feature selection in various research problems. They have been effectively utilized for feature selection across various datasets [13]. It is a swarm intelligence technique that is particularly built to handle continuous optimization challenges. Empirical data suggest that this algorithm outperforms or is on par with various state-of-the-art approaches [14]. The hunting technique of humpback, Whales has influenced the creation of WOA. Whales try to fill the new place in the search area by finding the optimal components in the group in the proposed technique. The suggested method of feature selection using nature-inspired optimizers has been shaped by its ability to solve optimization problems efficiently and effectively, achieving global optimization while circumventing the constraints of local optima. The utilization of techniques inspired by nature is observed by Chen et al. [15] and Abualigah et al. [16] to choose and optimize solutions.

The major objective of this work is to propose a new malware detection system based on Whale Optimization Algorithm (WOA) for feature selection and K-nearest neighbors (KNN) on the CIC-MalMem-2022 dataset. The proposed system for malware detection is designed to efficiently and accurately identify different malware variants. The core contribution of this paper is reducing the redundant, irrelevant features. Therefore, minimizing the feature set for the subset and simplifying the model's structure. Through experimental analysis and evaluation results, we will prove the efficiency of the proposed system and show its potential impact on enhancing cybersecurity resilience in malware detection.

The following sections of the paper are organized into clear parts. Section 2 gives an overview of the relevant literature. Section 3 describes the WOA and outlines the proposed model based on its principles. Section 4 presents the experimental analysis and results. Lastly, Section 5 wraps up with conclusions and recommendations for future research.

2. Related Literature

In this section, recent malware detection systems will be presented. All the malware detection systems that are discussed in this section use the same dataset.

Abualhaj et al. [17] improved the accuracy and efficiency of the malware detection system based on a K-nearest neighbor by modifying the distance metric. They have examined various distance measurements, such as Minkowski, Manhattan, and Euclidean metrics. Also, they used KNN to classify the captured samples for various aspects of similarity. The MalMem-2022 dataset was utilized to evaluate the efficiency of the proposed detection system. The outcomes showed good results with binary classifications. But for multi-class classification, KNN got better results on the Manhattan distance metric.

Damaševičius et al. [18] used many machine-learning techniques to evaluate malware detection by calculating the average precision, accuracy, and recall across 100 different random states. The author improved the accuracy and precision of the malware detection system by using random forest. However, the componential time for the proposed system is high.

Al Saaidah et al. [19] enhanced the performance of the detection malware system using the Firefly optimization algorithm KNN. Also, the authors presented an intelligence detection system for utilizing Machine Learning to Combat Ransomware Threats in Cloud Computing Environments. Genetic algorithms and particle swarm optimization techniques are employed to enhance the accuracy and efficiency of network traffic classification through effective feature selection. Then, the random forest technique was used for classification. The system has good accuracy in detecting ransomware attacks but needs to improve its performance [20].

Dener et al. [21] integrated machine learning and deep learning methods inside a structure for massive data to use memory data for virus detection. Using the CIC-MalMem 2022 dataset, the researchers ran trials and highlighted that the logistic regression approach had the highest best rate. When combined with the logistic regression method, the accuracy of the gradient-boosted tree approach is quite high—99.94%. In the field of malware analysis using memory data, the Naive Bayes method recorded the lowest accuracy at 98.41%.

Shafin et al. [3] investigated a novel method for detecting malware, that is both multiclass and lightweight, allowing it to identify contemporary infections and be suitable for execution on built-in equipment. The authors combined the feature extraction strengths of convolutional neural networks (CNNs) with the temporal modeling abilities of bidirectional long short-term memory (Bi-LSTM) networks. Comprehensive evaluations on the CIC-MalMem-2022 dataset show that this approach outperforms other machine learning techniques discussed in previous studies. The proposed solution offers a strong and streamlined model that can run on IoT devices to protect against obfuscated malware.

Alawad et al. [22] proposed a strategy to address the binary domain of feature selection in the IDS prediction model by improving the WSO technique. They create an initial population with notable diversity using two transfer functions and an altered K-means technique. There are three improved rounds of the binary WSO process that are suggested: BIWSO1, BIWSO2, and BIWSO3. The authors assess the effectiveness of these iterations utilizing a dozen publicly available real-world IDS and IoT datasets. An investigation is conducted in which the BIWSO3 is compared to established meta-heuristic techniques. The results demonstrate the effectiveness of the BIWSO3 strategy in terms of recall, accuracy, precision, and F1 metrics of categorization.

Although a range of optimization algorithms, such as particle swarm optimization (PSO) and genetic algorithm (GA), are commonly applied in feature selection, WOA was chosen for this study due to its efficacy in balancing exploration and exploitation during the optimization process. This balance is essential for selecting relevant features without sacrificing detection accuracy.

A comparative analysis with PSO and GA further highlights WOA's effectiveness, particularly in minimizing the number of selected features while maintaining high accuracy. This approach not only enhances the speed of malware detection but also reduces system demands, making it a suitable choice for real-time applications in cybersecurity.

One of the main challenges with PSO and GAs is finding the right balance between exploring new solutions and making the most of what's already known. They can also struggle with larger and more complex problems. Plus, their effectiveness can vary greatly depending on the specifics of the problem at hand. However, the WOA addresses these issues effectively. WOA often outperforms traditional methods like PSO and GAs, thanks to its unique way of mimicking natural behaviors, which helps it navigate complicated search spaces more efficiently. Then, we choose WOA after an in-depth comparison with PSO, and other optimization techniques according to time and accuracy. Then, we use WOA in an enhanced method because it outperforms other optimization techniques in the feature selection process [23].

By identifying the optimal set of features, this WOA-enhanced model effectively manages large datasets, detecting complex malware patterns with increased accuracy and lower rates of false positives and negatives. This capability is crucial for organizations aiming to stay ahead of emerging threats and implement proactive defense measures. Consequently, this study provides a valuable contribution to cybersecurity by introducing a streamlined and effective methodology for robust and precise malware detection.

3. Preliminaries

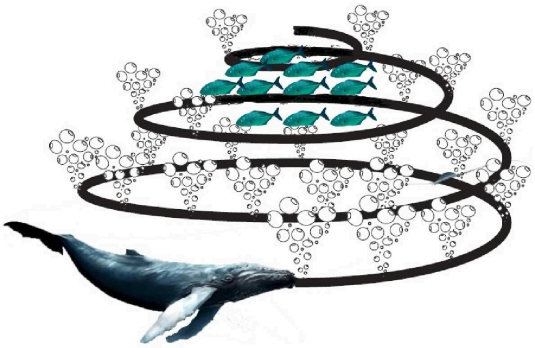
3.1. Whale Optimization Algorithm (WOA)

The WOA is a swarm intelligence technique inspired by the social behaviors observed in whale populations. Usually, these whales depend on their foraging behavior for survival. While various strategies have been developed to address optimization challenges, the whale algorithm stands out for its distinctive approach of using either a randomly chosen or optimal agent in the search space to target its prey. By employing spiral patterns, the model can replicate the bubble-net feeding strategies observed in humpback whales [23].

Specifically, humpback whales employ a distinctive foraging strategy characterized by the targeted pursuit of aggregations of small fish in proximity to the water's flatten. As depicted in Figure 1, the whales execute a discernible technique involving the creation of spiral bubble formations.

These spiraling bubbles serve to encircle and capture prey effectively. The foraging process encompasses the generation of cochleate bubbles surrounding the victim, synchronized with the whales' descent and ascent from the water's surface.

Figure 1
The spiral bubble-net attack strategy's schematic



This algorithm contains three operations that emulate the behavior of humpback whales through the steps of looking for the prey. The algorithm is designed using three operators that replicate the behavior of humpback whales throughout the stages of looking for victims (i.e., exploration technique). After that, enclosing victims by surrounding them and prey them (i.e., exploitation technique). The explanation of the mathematical model for WOA is mentioned in detail in the following pseudocode. More information can be found in Tan and Mohamad-Saleh [24]. Algorithm 1:

Algorithm 1: Pseudocode of the Whale Optimization Algorithm

```
Initialize the whales population $X_{i(i=1..2..3......n)}$
Compute the fitness of each whale.
Set $X^*$ as the best whale.
While (t < $maximumnumberofiterations$) do
for (eachsearchwhale) do
        Update a, $A$, $C$, $\iota$ and $p$.
        if ($p < 0.5$) then
           if ($|A| < 1$) then
                The whale position is updating by the
                Eq. (1).
          else
             if ($|A| \geq 1$) then
                 Select the random whale $X_{rand}$.
                 The whale position is updating by the
                 Eq. (9).
             end
          end
    else
        if ($p \geq 0.5$) then
          Modify the whale position by the
          Eq. (5).
             end
          end
    end
    Check if any search agent goes beyond the search
    Space and amend it.
    Compute the fitness of  each search agent.
    Update $X^*$ if there is a better solution.
    $t = t + 1$
    end
```

3.2. The proposed feature selection method

In this section, the steps for WOA will be explained for using it in the feature selection stage in the context of the malware detection system illustrated in Figure 2. The following subsections will go over each of the major stages in detail.

3.2.1. Dataset description

Obfuscated malware is a type of harmful software that uses different techniques to avoid being detected and removed. The main purpose of the obfuscated malware dataset is to test how well different malware detection methods work.

The CIC-Malmem-2022 dataset, released by the Canadian Institute for Cybersecurity, aims to advance academic research in malware classification, focusing specifically on techniques for identifying obfuscated malware. The dataset was designed to reflect real-world conditions by including common types of malware, such as Spyware, Ransomware, and Trojan Horses. Figure 3 shows memory dump categories. This dataset can be utilized to evaluate and enhance machine learning techniques for the detection of

Figure 2
Proposed model architecture

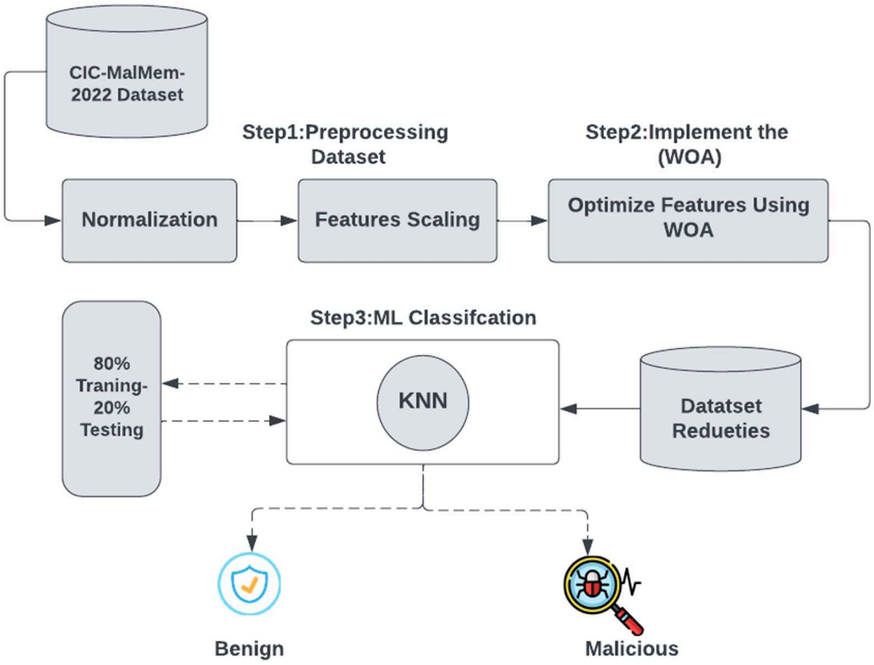
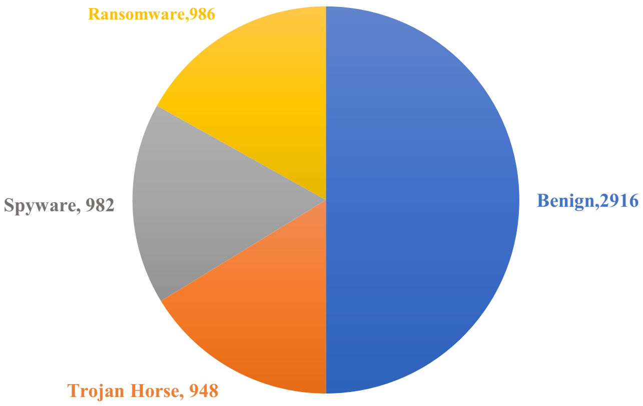


Figure 3
Memory dump categories



obfuscated malware, specifically those that rely on metaheuristic algorithms [25].

To expand the dataset, an automated method was employed, which involved executing 2,916 malware samples across three distinct categories using a virtual machine (VM). Acknowledging the importance of interspersing benign processes during the generating of malicious memory dumps, various applications were launched simultaneously within the Windows virtual machine alongside the execution of the malware samples. For each malware sample, 10 memory dumps were generated at intervals of 15 s, yielding 29,298 memory snapshots to comprehensively document possible malware activities [26]. For benign dumps, typical user behavior was recorded by activating various applications on the machine. The synthetic minority over-sampling technique (SMOTE) algorithm was employed to balance the dataset by generating synthetic values. In contrast to traditional oversampling methods, SMOTE creates new samples that closely resemble real values. This approach promotes a more balanced and comprehensive dataset, which is crucial for precise analysis and classification in future research.

The dataset is evenly balanced, with 50% of the entries being malicious and 50% benign memory dumps. In total, there are 58,596 entries, divided into 29,298 benign samples and 29,298 malicious ones. Figure 4 illustrates the breakdown of malware families in each category [25].

3.2.2. Preprocessing dataset

To uphold consistency and enhance the performance of optimization algorithms, the dataset underwent preprocessing as shown in the following points.

Normalization: Scaling numerical characteristics in the dataset is crucial to ensure all features are comparable. This process prevents any single feature with larger values from overshadowing others, which is vital for accurate data classification. To achieve this, Min-Max scaling was utilized, normalizing values to a uniform range of 0 to 1. This method ensures uniform and unbiased data analysis. To accomplish this, Min-Max normalization was employed, using the formula:

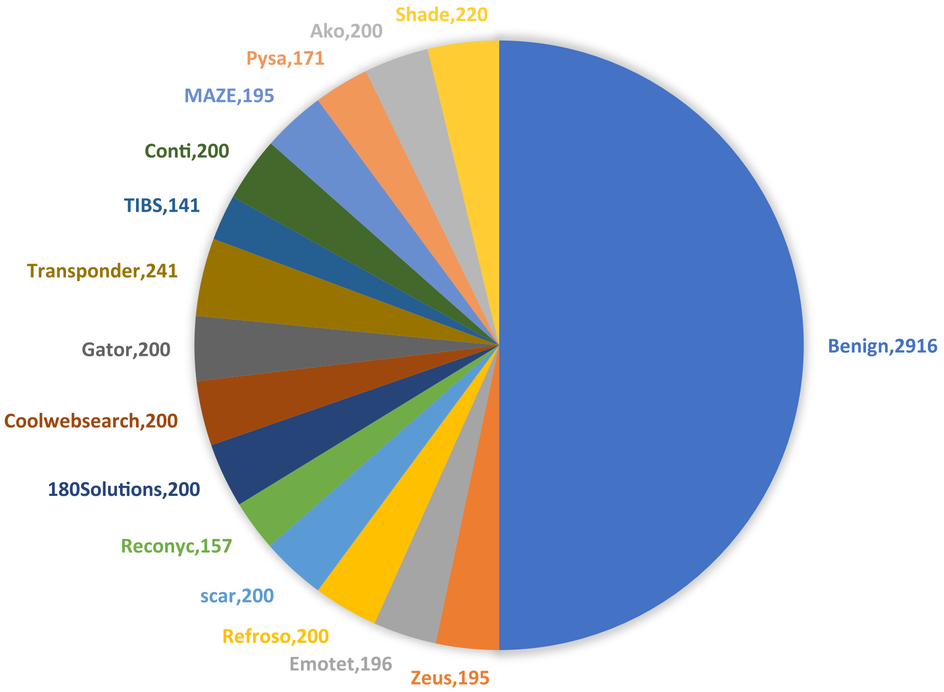
$$\textbf{Normalize Values} = \frac{\text{Original Value} - \text{Min}\,(\text{A})}{\text{Max}\,(\text{A}) - \text{Min}\,(\text{A})} \tag{1}$$

where the initial value denotes the original numerical entries of the dataset, *Min(A)* represents the smallest value observed for feature A in the dataset, and *Max(A)* represents the highest value attained by feature A within the dataset.

Features scaling: Rounding the numerical values in the CIC-Malmem-2022 dataset to four decimal places aims to improve generalization and mitigate overfitting, ensuring precise representation of numeric data.

Synthetic Minority Oversampling Technique (SMOTE): Given the natural imbalance in malware and benign samples, SMOTE was selected to address class imbalance by synthetically

Figure 4
Malware breakdown



generating new samples for minority classes. Instead of traditional oversampling, which can introduce biases by duplicating existing samples, SMOTE creates synthetic samples that interpolate between real instances, promoting a more realistic and diverse representation of minority class data. This method was chosen for its ability to provide a balanced dataset, which is essential for training robust classification models. By equalizing the number of benign and malicious samples, SMOTE enhances the classifier's ability to identify both classes accurately, improving model generalization and reducing the risk of bias in favor of the majority class.

3.2.3. Optimize features using WOA

WOA is employed for feature selection, which is a crucial step in enhancing the efficiency of the classification process. The central idea here is to optimize the feature set used for classification by the KNN classifier. KNN is a widely recognized classification technique where an unknown example is classified and determined by the most frequent class among its KNNs in the feature space [27]. In this paper, KNN serves as the classifier that assesses the classification performance of the selected feature subset. KNN is implemented to classify malware and benign software using the selected feature subset. The dataset is split with 80% allocated for training and 20% reserved for testing to assess the model's performance. The KNN classifier is then trained on the training data, using the optimized feature subset. The adaptive utilization of WOA in this paper, which aims to identify the most suitable feature subset that maximizes the classification performance. Within the framework of WOA, whales exhibit the ability to adapt their positions within a given search space. Initially, they tend to mimic the movements of the most successful search agent. Afterward, they strive to improve their rankings with the most effective search engine, as shown in the following equations.

$$\vec{D} = \vec{C}.\vec{X}^{*}(t) - X(t), \tag{2}$$

$$\vec{X} = (t+1)\vec{X}^{*}(t) - \vec{A}.\vec{D} \tag{3}$$

where t refers to the exiting iteration, $\vec{A}$ and $\vec{C}$ refer to coefficient vectors, $\vec{X}^{*}$ indicates the position of the optimal solution, and $\vec{X}$ refers to the vector's position. In case more than one optimal solution is determined, the variable $\vec{X}^{*}$ will be upgraded by iterative procedures.

Each solution in the dataset is expressed as a continuous vector with dimensions identical to those of the original feature space. The solution vector is bounded inside the interval [0, 1]. The fitness assessment values of these solutions are dichotomous, usually this reflects both the classification accuracy and the quantity of features chosen. This can be expressed by the following equation.

$$\mathbf{f\theta} = \alpha.\mathrm{E} + (1-\alpha)\,\frac{\sum_{i}^{N}\theta_i}{N} \tag{4}$$

where $\mathbf{f\theta}$ refers to the fitness function evaluated for a vector $\boldsymbol{\theta}$ of size N, θ_i refer to the binary elements indicating selected ($\theta_i = 1$) or unselected ($\theta_i = 0$) features. N represents the overall count of features present in the dataset. E denotes the classification error linked to the chosen subset of features. α is a constant parameter that governs the balance between the classification error and the number of selected features.

The WOA is employed to enhance feature selection, aiming to enhance classification accuracy by selecting the most informative subset of features. The following steps explain the proposed WOA algorithm.

Initialization of WOA parameters: The WOA parameters were initialized as follows:

1) Population Size: The algorithm was executed with a population size of whales, each representing a potential feature subset.
2) Maximum Iterations: We defined 30 iterations as the maximum number of times the algorithm would update and refine the feature subsets.

3) Exploration/Exploitation Factors: These factors were set to manage the exploration of new solutions versus the exploitation of promising ones, ensuring a balanced search through the feature space.

Define objective function: The objective function evaluates the classification accuracy using KNN with a given feature subset. Use a KNN classifier to assess the fitness of each whale (feature subset).

Initialization of population of whales: Initialized a population of whales, where each whale represented a various feature group. Feature groups were indicated to binary vectors (total number of features in the dataset), indicating the selection (1) or unselecting (0) of each feature. Randomly initialize the position of each whale in the search space.

Calculate fitness: For each whale in the population, calculate the fitness value (classification accuracy) using KNN classifier.

Find best position(X)***:** Identify the whale with the best fitness value. This position is denoted as $(X)^*$.

Update positions: The WOA executed iteratively to optimize feature subsets each iteration involved updating the positions (feature subsets) of whales within the search space, determining the best subset of features to enhance classification accuracy. Upon discovering a superior solution, the variable $(X)^*$ undergoes iterative updates.

Iterate: Repeat the fitness calculation and position update steps until the maximum number of iterations has been reached.

Select optimal position (X)***:** After completing the iterations, the position of the whale the feature subset exhibiting the greatest fitness value is chosen as the optimal one. Identified the feature subset that achieved the greatest classification accuracy using the KNN classifier. This subset was considered the optimized set of features, selected based on its superior performance in enhancing classification accuracy.

The integration of WOA into the feature selection process demonstrated its effectiveness in enhancing the performance of KNN classification by systematically exploring and exploiting feature subsets. This approach not only improved accuracy but also provided insights into the most relevant features for the classification.

4. Experimental Results and Discussion

Eventually, the experimental analysis and results demonstrate that the proposed system is both accurate and efficient. Various statistical metrics were used to highlight its effectiveness compared to other recently developed malware detection methods [28]. To confirm the efficiency of the presented method, the average number of features, efficiency, error rate, and accuracy have been compared among many feature selection methods. Table 1 shows the average number of features based on 30 runs.

Table 1
Average number of features associated with the optimization algorithms based on 30 runs

Algorithm	Average number of features
GOA	25.5
GA	24.2667
PSO	21.8667
ALO	13.6333
WOA	9.2333
BOA	24.9667
SMA	26.4

GOA algorithm has an average number of features at 25.5, indicating that it tends to use a slightly greater number of features compared to certain other algorithms in the list. GA shows an average of 24.2667 features, which is slightly lower than GOA, but still relatively high. It might imply a similar usage of features compared to GOA but with a slightly reduced number. PSO reflects an average of 21.8667 features. It seems to use fewer features on average compared to GOA and GA, potentially indicating a more focused or selective approach. ALO demonstrates a notably lower average number of features at 13.6333. This algorithm seems to operate with significantly fewer features compared to the previous ones, potentially emphasizing efficiency and targeted selection. WOA has even the lowest average of 9.2333 features, which indicates a distinct approach that relies on significantly fewer features than other algorithms in the list. BOA stands at an average of 24.9667 features, similar to GA, suggesting its similarity in feature utilization. SMA presents an average of 26.4 features. With a higher average number of features, suggests a comprehensive feature usage, similar to GOA, GA, and BOA. These results indicate diversity among these optimization algorithms in terms of their approach to utilizing features. Algorithms like ALO and WOA use notably fewer features on average, which might imply a more focused and efficient utilization of features, with an emphasis on reducing dimensionality and enhancing performance.

The following are the most features that are significantly influenced by malware behavior:

Handles average per process (handles.avg handle per proc): This metric refers to the average number of handles (e.g., file handles, registry handles) that each process on a system holds. It can indicate potential malware activity by revealing abnormal handle counts, which are often associated with malicious behavior such as excessive file manipulation, data exfiltration, resource exhaustion, and persistence mechanisms. Monitoring handle counts aids in detecting anomalies and profiling process behavior, providing insights into file-related activities of malware.

Process services (svcscan.process services): This feature tracks the services running on a system and their associated processes. It is crucial for understanding malware behavior as it can detect service hijacking, code injection into services, unauthorized service creation, privilege escalation, and anomalous behavior in service-associated processes. These indicators help identify potential malware activities and tactics used to gain persistence and evade detection.

Modules not loaded (ldrmodules.not in load): This feature refers to modules or DLLs not loaded through standard Windows mechanisms. It is significant for detecting code injection, fileless malware, stealth and evasion tactics, privilege escalation, and persistence mechanisms. Monitoring for nonstandard module loading helps uncover sophisticated malware strategies that bypass traditional detection methods.

Number of threads (handles.nthread): This metric tracks the number of threads a process has created. It provides insights into multithreaded malware behavior, including efficiency improvements, stealth tactics through concurrent execution, thread injection for evasion, resource consumption, and analysis evasion. Monitoring thread counts aids in detecting resource-intensive activities and understanding the complexity of malware.

Number of processes (pslist.nproc): This feature monitors the number of processes running on a system. It helps in identifying process enumeration tactics, persistence mechanisms, evasion of detection through process proliferation, resource consumption, and behavioral profiling. Unusual numbers of processes may indicate malicious activities like denial-of-service attacks or surreptitious process generation. Monitoring process counts over time aids in detecting deviations from normal patterns, indicating potential malware presence.

Table 2 shows the computational time based on 30 runs. The ALO and WOA present notably lower computational time, indicating higher efficiency. However, The BOA demonstrates noticeably extended computational times. This suggests that the process is relatively slower. The GOA, GA, and PSO methods exhibit moderately comparable computational durations. Therefore, WOA displays faster convergence, while the other algorithms demand significantly greater computational resources compared to this one.

Table 2
Average of computational time associated with the optimization algorithms based on 30 runs

Algorithm	Average time
GOA	65.9511
GA	65.2379
PSO	62.2649
ALO	37.7846
WOA	29.0548
BOA	232.9896
SMA	101.8269

Table 3 illustrates the average fitness values based on 30 runs. The GOA, GA, and BOA showcase similar average fitness values, implying comparable performance in optimizing fitness. In contrast, PSO reveals a slightly lower average fitness value compared to GOA, GA, and BOA, indicating a marginally high effective optimization. ALO and WOA present notably lower average fitness values, signifying a relatively higher overall performance in achieving optimal fitness. Notably, the SMA stands out with a higher average fitness value among the listed algorithms, indicating a relatively more effective performance in optimizing fitness as shown in Figure 5. That means, less fitness function reflects higher performance.

The fitness function calculation incorporates both the classification error rate and a minimal count of selected features, which may be expressed mathematically as:

$$\xrightarrow{Fitness} = \alpha\gamma\,(R) + \beta\frac{|F|}{|N|} \tag{5}$$

where $\gamma(R)$ refers to the classifier error rate, $|F|$ refers to the number of selected features, and |N| is the total number of features. $\alpha \in [0,1]$ and β = (1 − α) are two variables. Algorithm 1 also includes the pseudocode for the original WOE algorithm.

GOA, GA, PSO, and BOA methods exhibit notably high average accuracy values, implying a similar level of performance in achieving accurate solutions as shown in Table 4. However, ALO and WOA display slightly lower average accuracy values,

Table 3
Average of fitness values associated with the optimization algorithms based on 30 runs

Algorithm	Average fitness
GOA	0.0050531
GA	0.0049528
PSO	0.0044488
ALO	0.0033179
WOA	0.0025686
BOA	0.0050181
SMA	0.0052618

Table 4
Average of accuracy associated with the optimization algorithms based on 30 runs

Algorithm	Average accuracy
GOA	0.99958
GA	0.99945
PSO	0.99952
ALO	0.99915
WOA	0.99910
BOA	0.99952
SMA	0.99953

Figure 5
Comparison of fitness values

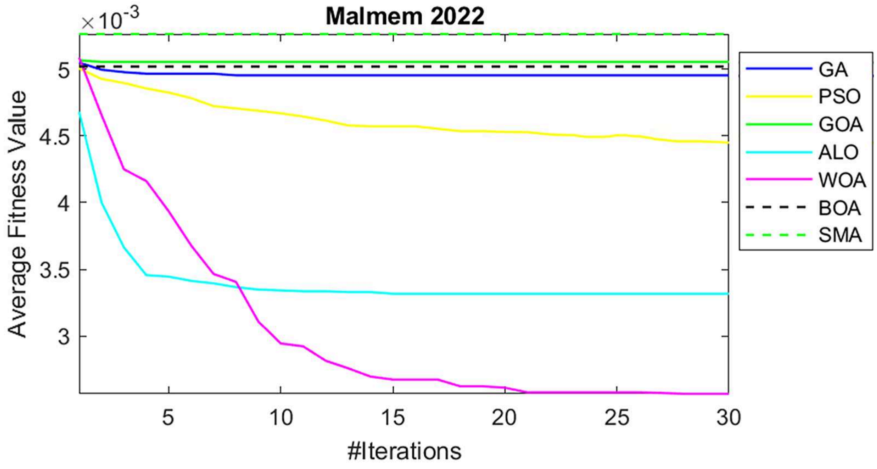
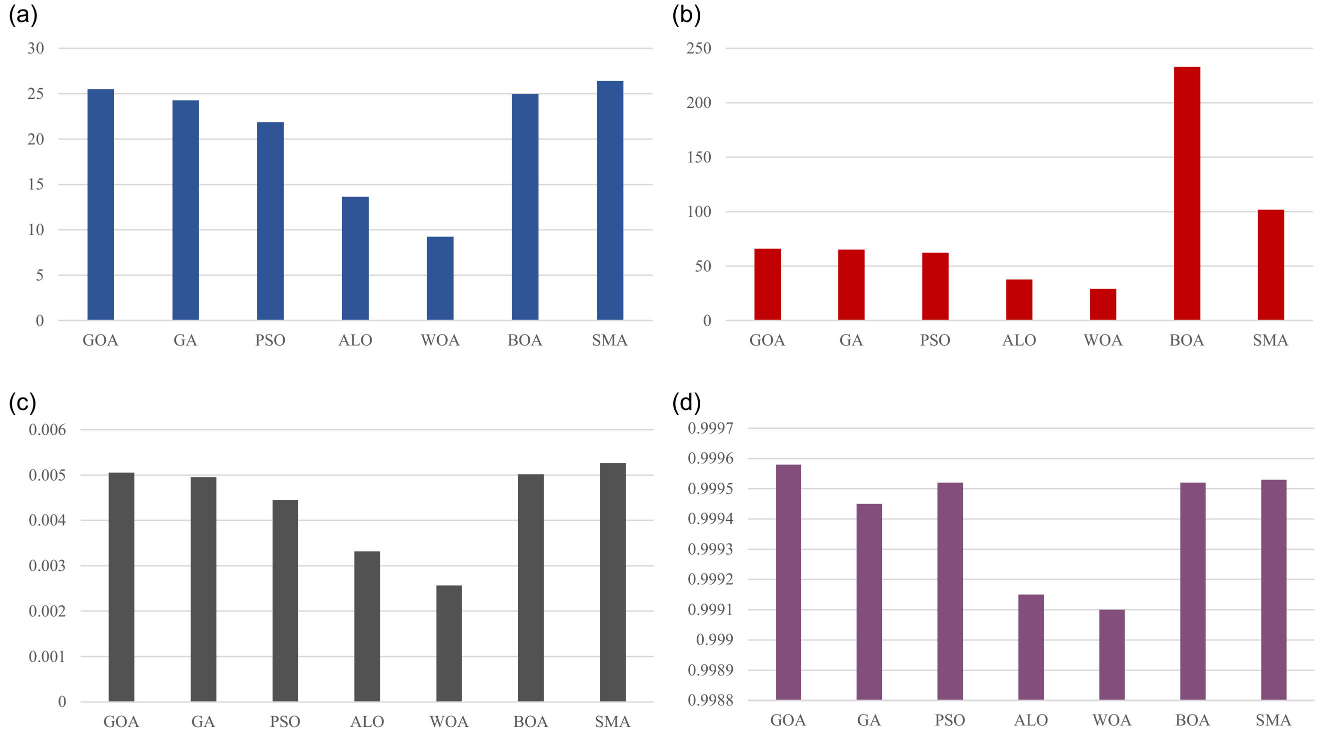


Figure 6
A comparative of optimization algorithms performance in terms of (a) features, (b) computational time, (c) fitness values, and (d) accuracy based on an average of 30 runs



suggesting a relatively lower overall accuracy in achieving optimal solutions. Conversely, the SMA stands out with a higher average accuracy value among the listed algorithms, indicating a relatively more effective performance in achieving accurate solutions.

A wide analysis of optimization algorithms, considering the number of features, fitness values, accuracy, and computational efficiency, provides valuable insights into the strengths and trade-offs of each. Notably, the WOA algorithm displays an efficient and streamlined approach with lower feature usage, superior fitness values, high accuracy, and quicker convergence. However, to illustrate these results, a chart in Figure 6 provides a clear and concise representation of the comparative performance of each algorithm across the average of 30 runs on the four-evaluation metrics.

5. Conclusion

High dimensionality is a common challenge in machine learning that complicates and slows down malware detection methods. This paper introduces a new detection method that combines the WOA with the KNN machine learning technique. WOA addresses the issues of high dimensionality by eliminating irrelevant and redundant features. The experimental results indicate that this approach achieves near-ideal performance across several metrics, such as the average number of features, accuracy, and processing time. Notably, the number of features was reduced from 55 to just 9.23 on average, while still demonstrating strong effectiveness in identifying malware. To our knowledge, this is the first study to use feature selection to enhance malware detection on the CIC-MalMem-2022 dataset. Looking ahead, we plan to improve the method to capture a wider range of malware characteristics, including API usage. Additionally, we will enhance the WOA search engine to better handle optimization challenges and avoid getting stuck in local optima. There is also the potential for applying WOA to other diverse datasets, which warrants further exploration.

Conflicts of Interest

The authors declare that they have no conflicts of interest to this work.

Data Availability Statement

Data are available on request from the corresponding author upon reasonable request.

Author Contribution Statement

Mariam Al Ghamri: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing – original draft, Writing – review & editing, Supervision, Project administration. **Dyala Ibrahim:** Conceptualization, Validation, Formal analysis, Writing – original draft, Writing – review & editing, Project administration. **Rami Sihwail:** Methodology, Validation, Data curation, Writing – original draft, Writing – review & editing. **Mohammad Shehab:** Methodology, Software, Validation, Investigation, Data curation, Writing – original draft, Writing – review & editing, Visualization.

References

[1] Sihwail, R., Omar, K., & Ariffin, K. A. Z. (2018). A survey on malware analysis techniques: Static, dynamic, hybrid and

memory analysis. *International Journal on Advanced Science, Engineering and Information Technology, 8*(4–2), 1662–1671. https://doi.org/10.18517/ijaseit.8.4-2.6827

[2] Jerbi, M., Dagdia, Z. C., Bechikh, S., & Said, L. B. (2023). Immune-based system to enhance malware detection. In *IEEE Congress on Evolutionary Computation*, 1–8. https://doi.org/10.1109/cec53210.2023.10254159

[3] Shafin, S. S., Karmakar, G., & Mareels, I. (2023). Obfuscated memory malware detection in resource-constrained IoT devices for smart city applications. *Sensors, 23*(11), 5348. https://doi.org/10.3390/s23115348

[4] Aslan, O., & Samet, R. (2020). A comprehensive review on malware detection approaches. *IEEE Access, 8,* 6249–6271. https://doi.org/10.1109/ACCESS.2019.2963724

[5] Li, Y., Xiong, K., Chin, T., & Hu, C. (2019). A machine learning framework for domain generation algorithm-based malware detection. *IEEE Access, 7,* 32765–32782. https://doi.org/10.1109/access.2019.2891588

[6] Gaber, M. G., Ahmed, M., & Janicke, H. (2024). Malware detection with artificial intelligence: A systematic literature review. *ACM Computing Surveys, 56*(6), 148. https://doi.org/10.1145/3638552

[7] Oyewole, O. O., Fakeyede, O. G., Okeleke, E. C., Apeh, A. J., & Adaramodu, O. R. (2023). Security considerations and guidelines for augmented reality implementation in corporate environments. *Computer Science & IT Research Journal, 4*(2), 69–84. https://doi.org/10.51594/csitrj.v4i2.607

[8] Ahmad, R. W., Salah, K., Jayaraman, R., Yaqoob, I., Ellahham, S., & Omar, M. (2021). The role of blockchain technology in telehealth and telemedicine. *International Journal of Medical Informatics, 148,* 104399. https://doi.org/10.1016/j.ijmedinf.2021.104399

[9] Tang, M., Alazab, M., & Luo, Y. (2019). Big data for cybersecurity: Vulnerability disclosure trends and dependencies. *IEEE Transactions on Big Data, 5*(3), 317–329. https://doi.org/10.1109/tbdata.2017.2723570

[10] Gibert, D., Mateu, C., & Planes, J. (2019). A hierarchical convolutional neural network for malware classification. In *International Joint Conference on Neural Networks,* 1–8. https://doi.org/10.1109/ijcnn.2019.8852469

[11] Wang, Q., & Qian, Q. (2022). Malicious code classification based on opcode sequences and text CNN network. *Journal of Information Security and Applications, 67,* 103151. https://doi.org/10.1016/j.jisa.2022.103151

[12] Sihwail, R., Omar, K., Ariffin, K. A. Z., & Tubishat, M. (2020). Improved Harris hawks optimization using elite opposition-based learning and novel search mechanism for feature selection. *IEEE Access, 8,* 121127–121145. https://doi.org/10.1109/access.2020.3006473

[13] Kundu, R., Chattopadhyay, S., Cuevas, E., & Sarkar, R. (2022). AltWOA: Altruistic Whale Optimization Algorithm for feature selection on microarray datasets. *Computers in Biology and Medicine, 144,* 105349. https://doi.org/10.1016/j.compbiomed.2022.105349

[14] Amiriebrahimabadi, M., & Mansouri, N. (2024). A comprehensive survey of feature selection techniques based on Whale Optimization Algorithm. *Multimedia Tools and Applications, 83,* 47775–47846. https://doi.org/10.1007/s11042-023-17329-y

[15] Chen, G., Qin, Q., Ping, Z., Peng, K., Zeng, X., Long, H., & Zou, M. (2021). A novel approach based on modified and hybrid flower pollination algorithm to solve multi-objective optimal power flow. *IAENG International Journal of Applied Mathematics, 51*(4), 1–18.

[16] Abualigah, L., Al-Zyod, M., Ikotun, A. M., Shehab, M., Otair, M., Ezugwu, A. E., . . . , & El-kenawy, E.-S. M. (2024). A review of krill herd algorithm: Optimization and its applications. In L. Abualigah (Ed.), *Metaheuristic optimization algorithms: Optimizers, analysis, and applications* (pp. 231–239). Morgan Kaufmann. https://doi.org/10.1016/B978-0-443-13925-3.00017-0

[17] Abualhaj, M. M., Abu-Shareha, A. A., Shambour, Q. Y., Alsaaidah, A., Al-Khatib, S. N., & Anbar, M. (2024). Customized K-nearest neighbors' algorithm for malware detection. *International Journal of Data and Network Science, 8*(1), 431–438. https://doi.org/10.5267/j.ijdns.2023.9.012

[18] Damaševičius, R., Venčauskas, A., Toldinas, J., & Grigaliūnas, S. (2021). Ensemble-based classification using neural networks and machine learning models for Windows PE malware detection. *Electronics, 10*(4), 485. https://doi.org/10.3390/electronics10040485

[19] Al Saaidah, A., Abualhaj, M. M., Shambour, Q. Y., Abu-Shareha, A. A., Abualigah, L., Al-Khatib, S. N., & Alraba'nah, Y. H. (2025). Enhancing malware detection performance: Leveraging K-Nearest Neighbors with Firefly Optimization Algorithm. *Multimedia Tools and Applications, 84*, 10071–10094. https://doi.org/10.1007/s11042-024-18914-5

[20] Ghazi, M. R., & Raghava, N. S. (2022). Detecting ransomware attacks in cloud environment using machine learning-based intelligence system in COVID-19 chaos. In *IEEE Conference on Interdisciplinary Approaches in Technology and Management for Social Innovation,* 1–6. https://doi.org/10.1109/iatmsi56455.2022.10119441

[21] Dener, M., Ok, G., & Orman, A. (2022). Malware detection using memory analysis data in big data environment. *Applied Sciences, 12*(17), 8604. https://doi.org/10.3390/app12178604

[22] Alawad, N. A., Abed-Alguni, B. H., Al-Betar, M. A., & Jaradat, A. (2023). Binary improved white shark algorithm for intrusion detection systems. *Neural Computing and Applications, 35*(26), 19427–19451. https://doi.org/10.1007/s00521-023-08772-x

[23] Nadimi-Shahraki, M. H., Zamani, H., Asghari Varzaneh, Z., & Mirjalili, S. (2023). A systematic review of the whale optimization algorithm: Theoretical foundation, improvements, and hybridizations. *Archives of Computational Methods in Engineering, 30*(7), 4113–4159. https://doi.org/10.1007/s11831-023-09928-7

[24] Tan, W.-H., & Mohamad-Saleh, J. (2023). A hybrid whale optimization algorithm based on equilibrium concept. *Alexandria Engineering Journal, 68,* 763–786. https://doi.org/10.1016/j.aej.2022.12.019

[25] Carrier, T., Victor, P., Tekeoglu, A., & Lashkari, A. (2022). Detecting obfuscated malware using memory feature engineering. In *Proceedings of the 8th International Conference on Information Systems Security and Privacy, 1,* 177–188. https://doi.org/10.5220/0010908200003120

[26] Hossain, A., & Islam, S. (2024). Enhanced detection of obfuscated malware in memory dumps: A machine learning approach for advanced cybersecurity. *Cybersecurity, 7*(1), 16. https://doi.org/10.1186/s42400-024-00205-z

[27] Li, A.-D., Xue, B., & Zhang, M. (2021). Improved binary particle swarm optimization for feature selection with new initialization and search space reduction strategies. *Applied Soft Computing, 106,* 107302. https://doi.org/10.1016/j.asoc.2021.107302

[28] Smith, D., Khorsandroo, S., & Roy, K. (2023). Supervised and unsupervised learning techniques utilizing malware datasets. In *IEEE 2nd International Conference on AI in Cybersecurity,* 1–7. https://doi.org/10.1109/icaic57335.2023.10044169

How to Cite: Al Ghamri, M., Ibrahim, D., Sihwail, R., & Shehab, M. (2025). Whale Optimization Algorithm for Feature Selection Enhances Classification in Malware Datasets. *Journal of Computational and Cognitive Engineering*, *4*(3), 387–396. https://doi.org/10.47852/bonviewJCCE42024233