Received: March 5, 2024. Revised: April 6, 2024.

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

An Enhanced Model of Whale Optimization Algorithm and K-nearest Neighbors for Malware Detection

Rami Sihwail[1]* **Mariam Al Ghamri[1]** **Dyala Ibrahim[1]**

[1]*Department of Cyber Security, College of Computer Science and Informatics, Amman Arab University, Amman, Jordan*
* Corresponding author's Email: r.sihwail@aau.edu.jo

Abstract: The threat of malicious software has evolved into a major concern regarding the security of the system and network infrastructure. Machine learning algorithms have been successfully utilized to classify malware files into malicious or benign. However, the exponential growth in data volume and feature dimensionality poses challenges for machine learning, resulting in reduced classification accuracy and heightened computational costs. Feature selection is an essential process that can address these challenges by eliminating irrelevant, redundant, and less informative features that may adversely affect classifier performance. In this study, we introduce an enhanced Whale Optimization Algorithm (EWOA) aimed at improving classification accuracy, feature selection, and overall malware detection model efficiency. The proposed EWOA introduces an enhanced search mechanism that integrates mutation and neighborhood search strategies, aiming to refine its exploration strategy. This novel approach is more adept at steering clear of local optima. Additionally, EWOA augments its population diversity by incorporating the Opposite-Based Learning technique (OBL) during its initial phase. To assess the efficacy of the proposed method, performance evaluations were conducted using the CIC-MalMem-2022 dataset. Various aspects including the number of features, efficiency, fitness value, accuracy, and statistical tests were compared across different optimization algorithms: Gray Wolf Optimization Algorithm (GOA), Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Artificial Lion Optimization (ALO), Butterfly Optimization Algorithm (BOA), and Slime Mould Algorithm (SMA). The experimental results affirm the superiority of EWOA over other optimization algorithms in diverse areas, such as classification accuracy (99.987%), fitness value (0.00084511%), and average feature count (on average, 3.97 features).

Keywords: CIC-MalMem-2022 dataset, Feature selection, KNN, Malware detection, Whale optimization algorithm (WOA).

1. Introduction

Malicious software is a program intentionally created to penetrate or cause damage to a computer system without the user's consent, leading to serious security issues [1]. Malware can be found on all platforms, including operating systems, custom builds, and mobile platforms. Malware, or malicious code, aims to damage, disrupt, steal, or participate in unlawful activities against information, hosts, or systems. There are different types of malware, some of which attach themselves to legitimate software or are attached to files as macros. Some exploit vulnerabilities in custom software or in the operating system itself. Network devices are also susceptible to attacks due to misconfigurations, such as a browser vulnerability that forces users to visit a website. Users' actions, such as opening emails from untrusted sources, browsing websites with malicious code, downloading programs from untrusted websites, and clicking on advertisements, are the main drawbacks that can be exploited to spread this malware. Typical forms of malicious software encompass viruses, worms, Trojans, bots, backdoors, spyware, and adware. Emerging viruses and polymorphic viruses employing obfuscation methods pose greater challenges for detection. One of the prevalent methods for malware detection software today is

Received: March 5, 2024. Revised: April 6, 2024.

using behavioral and signature-based detection techniques [2].

Signatures, which are concise sequences of bytes unique to individual programs [3]. Demonstrate a low error rate and are adept at identifying particular viruses within executable files, boot records, or memory [4]. Nevertheless, the signature-based approach proves ineffective when confronted with altered or unfamiliar malicious executables. Conversely, behavior-based detection holds promise in safeguarding against emerging and unforeseen threats. Nonetheless, this method of virus detection typically entails considerable costs and time investments [5].

Alternatively, memory-based techniques offer a unique approach to malware detection that differs from traditional static signature-based methods. Unlike static signatures, which are based on predefined patterns of known malware, memory-based techniques focus on analyzing the behavior and characteristics of a program as it executes. This dynamic approach creates an adaptive and proactive detection mechanism that identifies previously unknown or evolving malware threats. Utilizing the CIC-MalMem-2022 dataset extracted through memory analysis, these methods demonstrate their effectiveness in identifying polymorphic and metamorphic malware. In addition, memory-based techniques help to reduce false positives [6]. This approach increases the accuracy of malware detection systems by minimizing the risk of falsely identifying legitimate programs as malicious.

The original program was initially developed as an anti-malware system. Its purpose was to prevent viruses and Trojans from making unauthorized changes to files. John McAfee introduced his VirusScan™ software in 1989, which was able to detect and eliminate multiple viruses simultaneously [3]. These detection systems are statistical anomaly detectors that were developed by Los Access in 1989. They can detect anomalies based on statistical analysis. The problem with such programs is that they can be called up remotely by adversaries, enabling them to launch attacks without authentication. A Trojan horse is a type of code that masquerades as a useful program but is designed to either steal information or corrupt data [7]. Sniffers are software applications that are capable of intercepting and logging network traffic. They intercept each packet to decrypt and obtain the unprocessed data by revealing the values of the various components within the packet and examining its contents. Sniffer code can serve as the first step in an intrusion attempt. Spam, also known as junk email, is a software program that sends identical messages via email to a large number of specific recipients. Spam can cause system delays due to the influx of numerous emails, and it can also lead to bandwidth consumption. It is also occasionally used as a replacement for adware [7].

Software companies develop detection systems to analyze and categorize new programs. Valid software is added to a white list, while malicious software is added to a black list. To classify undecidable software, also known as grey list, scanners are used in a controlled environment [8]. If new malware is discovered during the analysis of a program on the gray list, the company issues online updates to remove the newly identified malware. Users can update their product databases remotely via an internet connection.

Most malware scanners use signature-based (static) and behavior-based (dynamic) methods to identify programs. However, there are different approaches to solve this problem. The primary distinction between signature-based and behavior-based techniques resides in the utilization of classifications for malware detection rather than depending on patterns [9].

The efficiency of signature-based and behavior-based techniques was improved through the utilization of machine learning (ML). However, both techniques are affected by the high dimensionality problem caused by the huge number of extracted features [10].

Feature selection is an important technique in ML because it is the most effective tool for solving the problem of high dimensionality. Due to the existence of irrelevant, redundant features, which are affected negatively on the accuracy of the final classifier. So, feature selection is important to select the optimal subset of features in the feature space. To reduce dimensionality, increase accuracy, and decrease runtime [11]. Optimization is one of the feature selection methods, which is used to select the optimal features from the feature space. Therefore, it reduces the dimensionality and increases the accuracy of classification. The optimal feature subset logically has the fewest features and the highest classification accuracy [12].

The Whale Optimization Algorithm (WOA) was improved in this study to increase the efficiency of feature selection. WOA is a proposed swarm intelligence approach specifically designed to tackle continuous optimization tasks. Empirical evidence has shown that WOA has superior or equivalent performance compared to several established algorithmic methods [13]. WOA was developed based on the hunting behavior of humpback whales. In an enhanced iteration of the WOA, a whale endeavors to shift to a new location within the search space by considering the optimal element in the group. The efficiency and ability of feature selection through nature-inspired optimizers to transform optimization problems into global optimization, thereby surpassing the constraint of local optima, has propelled their progress. Nature-inspired methodologies are employed for the purpose of choosing and refining solutions [14]. The primary constraints of this algorithm are its convergence speed and tendency to quickly fall into local optima [15, 16].

This paper proposes a novel malware detection model by utilizing the K-Nearest Neighbors (KNN) using and an enhanced version of WOA (EWOA) for feature selection. We propose a robust and adaptive model that is capable of identifying malware variants. By reducing the number of unnecessary features, malware detection systems can be made more accurate and efficient. The analysis results are demonstrated through rigorous experiments and evaluation of the proposed method, showing that it has the potential to improve cybersecurity in malware detection.

With technological progress, the number of malware is also constantly increasing. Malware has evolved to have transformative characteristics, leading to a significant increase in the diversity of malware types [17]. Furthermore, even inexperienced malware developers are now able to easily generate novel forms of malware thanks to automated malware development tools [18]. Traditional methods of identifying malware by characteristics have proven ineffective against the wide range of new malware [17]. However, ML strategies for malware detection have proven to be highly effective in detecting new malware. ML strategies for malware detection have a significant false positive rate in parallel [19]. Our goal is to maximize the detection rate with ML algorithms.

In addition, malicious programs employ diverse methods to disseminate viruses, seize control of computer systems and IoT devices, and pilfer sensitive data like credit card numbers and other personal information. Although there are many approaches to detecting intruders, identifying malicious code remains difficult. Current malware detection systems lack the ability to detect new threats and malware disguised as harmless programs. Because they rely on fixed collections of known malware cases, they are vulnerable to innovative and previously unknown dangerous behaviors. One of the main reasons for these challenges is the explosion in the number of features included in malware samples [4]. This proliferation of features brings with it problems related to the high-dimensionality of features, sparsity, and complexity of high-dimensional data, which are known challenges in data analysis. In addition, high-dimensional data negatively affects the performance of ML classifiers, causing a decrease in classification accuracy and an increase in computational costs [20].

The feature selection phase within a malware detection system tackles these hurdles by employing diverse optimization algorithms. Nonetheless, such optimization algorithms, like WOA, face challenges regarding convergence speed and a predisposition to rapidly converge towards local optima [21, 22]. Hence, EWOA was introduced to mitigate these constraints. By incorporating the proposed search strategies (SS), this extension seeks to improve the algorithm's effectiveness in exploring both global and local areas. Furthermore, this enhancement is anticipated to streamline the suggested malware detection model by reducing the quantity of chosen features, thereby enhancing its classification efficacy.

The main challenges associated with WOA, which include the speed of convergence and the restriction to local optima, arise from the following factors [21, 22]:

- Exploration vs. Exploitation: The fundamental challenge of striking a balance between exploration and exploitation represents an important aspect in optimization algorithms. Frequently, prioritizing exploration within the Whale Optimization Algorithm (WOA) may result in a thorough exploration of the search space, thereby postponing the convergence towards a promising solution.
- Random Movements: WOA employs various strategies inspired by the behavior of humpback whales, such as encircling prey, bubble-net feeding, and tail flicking. It is important to note that the implementation of these strategies involves some level of randomness that is likely to result in slower convergence, especially if random behavior is not guided correctly.
- High-Dimensional Spaces: In high-dimensional spaces, the optimization search space becomes more complex to navigate effectively. WOA's random movements and strategies might not perform optimally in such spaces, leading to difficulties escaping local optima.

This challenge can be overcome by enhancing the convergence speed of the WOA and addressing its limitations related to local optimum. Through the adoption of OBL techniques, the proposed EWOA enhances the initial population of whales. Moreover, the new search strategy enhances the search technique. Consequently, reducing the complexity of feature selection in malware detection can strengthen the accuracy and efficiency of identifying malware instances.

Finally, this study employs a range of evaluation metrics to evaluate the efficacy and efficiency of an enhanced EWOA. The CIC-MalMem-2022 dataset was assessed using evaluation measures such as the average number of features, efficiency, fitness value, accuracy, and statistical tests. The proposed method's results were compared with optimization algorithms such as WOA, GOA, GA, PSO, ALO, BOA, and SMA.

This paper is structured into distinct sections. Section 2 provides an overview of related works. Section 3 discusses the WOA. Section 4 outlines the enhancements to the proposed EWOA. Section 5 introduces the experimental analysis and presents the obtained results. Finally, Section 6 contains conclusions and future work.

2. Related work

Talukder and colleagues (2023) proposed a novel approach that integrates machine learning and deep learning methodologies with the aim of bolstering the efficacy of network intrusion detection systems. This integration yields augmented detection rates and heightened system reliability. Their method adeptly preprocesses data through a fusion of SMOTE for data balancing and XGBoost for feature selection. Comparative evaluations against alternative detection mechanisms illustrate the superior performance of the proposed method, showcasing remarkable outcomes across the KDDCUP'99 and CIC-MalMem-2022 datasets. However, this method relies heavily on specific datasets and can be computationally complex. [23].

Smith et al. (2023) utilized the Malware-exploratory and CIC-MalMem-2022 datasets for their investigation. The research involved the application of three clustering methods: K-Means, DBSCAN, and Gaussian Mixture Model (GMM). Seven classification techniques were employed within the model to forecast occurrences of malware: Decision Tree, Random Forest, Ada Boost, K-Neighbors, Stochastic Gradient Descent, Extra Trees, and Gaussian Naïve Bayes. The Malware Exploratory dataset achieved an average accuracy rate of 90%, whereas the CIC-MalMem-2022 dataset exhibited an average accuracy rate of 99%. Consistency in performance across all three clustering techniques was observed for both datasets. [17].

In their next research, Smith et al. (2023) used two different datasets, namely Malware-exploratory and CIC-MalMem-2022, to collect data for analysis using a range of supervised and unsupervised learning methods. This study builds on previous research by incorporating feature selection methods, including Pearson correlation coefficient and GA, into the developed model. The proposed model is then evaluated on a custom dataset called SMITH as well as a dataset created using a Generative Adversarial Network (GAN) trained on SMITH. The results of this study show that the genetic algorithm demonstrates a considerable level of proficiency in identifying malicious software in the Malware-Exploratory and CIC-MalMem-2022 datasets. On the other hand, the use of the Pearson correlation coefficient appears to be efficient when applied to the SMITH dataset [24]. However, their technique face challenges with generalizability and interpretability.

In addition, Shafin et al. (2023) proposed a novel approach to malware identification that has two important characteristics: Multiclass capability and lightweight. This approach enables the detection of modern malware while being compatible with embedded devices. This study suggests a hybrid model that merges the feature learning abilities of convolutional neural networks (CNNs) with the temporal modeling benefits of bidirectional long- and short-term memory (LSTM). Extensive experiments conducted on the CIC-MalMem-2022 dataset demonstrate the superior performance of this strategy [25].

Moreover, Jerbi et al. (2023) have proposed a method for malware detection, which contains two different procedures: the first procedure uses a memetic algorithm to generate new instances of malware, and the second procedure uses robust detectors generated by an artificial immune system based algorithm to identify these new instances of attacks. The effectiveness of a novel malware detection system has been demonstrated through extensive experimentation with heavy-duty datasets and evaluation criteria. However, the work still lack of real-world validation and interpretability [4].

Furthermore, Alawad et al. (2023) suggested an enhancing the White Shark Optimizer (WSO) technique to address feature selection in the binary domain of an Intrusion Detection System (IDS) prediction model. They employed two transfer functions and a customized K-means algorithm to create an initial population with substantial diversity. Three enhanced versions, BIWSO1, BIWSO2, and BIWSO3, were suggested to enhance the binary WSO procedure. The results indicate that the BIWSO3 method is successful in improving classification accuracy, precision, recall, and F1-measures [18]. Though, the technique may encounter issues with the implementation complexity and the sensitivity of parameter selection.

Dener et al. (2022) presented a novel approach to improve malware detection by leveraging memory data. The method they proposed involves the integration of deep learning and machine learning techniques into a comprehensive framework tailored for handling huge datasets. The researchers conducted experiments on the CIC-MalMem2022 dataset and found that the logistic regression technique showed the most success. The gradient boosted tree algorithm in combination with the logistic regression technique shows a remarkably high accuracy, especially with a hit rate of 99.94%. In the area of malware analysis using memory data, the Naive Bayes technique achieved the lowest accuracy of 98.41%. In addition, a significant portion of the algorithms used showed a remarkable level of performance. The results of this study show that the data obtained through memory analysis is of great value in identifying and detecting malware. Nevertheless, the work is likely to encounter challenges regarding generalization and overfitting [26].

Luhr and Hallqvist (2022) conducted a comparative analysis of a deep learner multi-layer perceptron (MLP) and an ensemble learner from traditional ML techniques. The comparison focused on evaluating the accuracy and runtime performance of these models. The dataset used in this study consists of obfuscation-based feature extraction data obtained from the volatile memory of computer systems infected by malware. The results of the study show that the MLP reduced classification times for binary malware by 94.3% compared to ensemble learning, with accuracy decreasing by only 0.02 percentage points. The 99.8% reduction in classification times for multi-class classification is associated with a 3.2 percentage point loss in accuracy. The results suggest that the MLP is a good choice for this particular task in practice, as it offers a significant improvement in time efficiency [27]. However, this work may overlook broader evaluation metrics and the generalizability of their findings.

Memory analysis plays a central role in the identification of malignant processes and enables the detection of various characteristics and behaviors Carrier (2021). Despite the extensive research in this area, there are still major challenges in malware detection, e.g. in terms of detection rates and the sophisticated obfuscation techniques used by advanced malware. As advanced malware uses obfuscation and other evasion techniques to evade conventional detection methods, this study seeks to extend VolMemLyzer, a state-of-the-art memory feature extractor for learning systems, with an increased focus on hidden and obfuscated malware. The extension includes the integration of the tool into a stacked ensemble of machine learning models, creating a robust framework for efficient malware detection. In addition, a specific malware memory dataset, namely CIC-MalMem-2022, has been carefully created to rigorously test and evaluate the proposed framework [28]. Table 1 presents a summary of the performance of previous studies.

As can be seen, the related studies mainly focused on using deep learning and machine learning to detect malware and suggested various techniques to improve detection rates and system reliability. Studies emphasize the importance of adapting to new and emerging malware threats, especially in the context of advanced technology and sophisticated attacks. However, previous studies did not attempt to reduce the number of selected features. Consequently, the complexity of their proposed systems is high. Therefore, we conducted a research study involving a malware detection model using a KNN classifier.

The model involves enhancing the WOA, with a particular focus on reducing the number of feature selections to reduce complexity, increase accuracy, and, therefore, increase the speed of malware detection. The results of this study can greatly contribute to the field of cybersecurity, providing valuable insights and methodologies for more robust and accurate malware detection systems.

Table 1. Show the summary of performance for relevant previous studies.

Author	**Methodology**	**Result (Accuracy)**
[23]	SMOTE, XGBoost	99.99%
[17]	machine learning algorithms	90%, 99%
[24]	Pearson correlation coefficient and GA.	90%, 99%
[25]	(CNNs), (LSTM)	-
[18]	(BIWSO)	BIWSO3 exhibits high efficiency
[26]	Logistic Regression Gradient Boosted Tree Naive Bayes	99.97% 99.94% 98.41%
[27]	(MLP), Ensemble Learner	reduced binary classification time by 94.3% and multiclass by 99.8%,
[28]	KNN, SVM	0.95% 0.90%

Received: March 5, 2024. Revised: April 6, 2024.

3. Preliminaries

Metaheuristic algorithms have gained significant attention in recent years for their ability to efficiently solve complex optimization problems. Numerous studies have utilized various metaheuristic approaches and have achieved notable results in solving a diverse range of optimization problems. These metaheuristics are often developed through the construction from scratch, modification of existing algorithms, or hybridization of multiple approaches. In this review, we highlight several recent studies that introduce novel metaheuristic algorithms and evaluate their performance against established methods.

It is worth mentioning that there are several related works that have significant improvements using metaheuristic optimization algorithms, including the Extended Stochastic Coati Optimizer (ESCO), Swarm Bipolar Algorithm (SBA), Swarm Space Hopping Algorithm (SSHA), Migration-Crossover Algorithm (MCA), Total Interaction Algorithm (TIA), Four Directed Search Algorithm (FDSA), and Attack Leave Optimizer (ALO). These algorithms have shown superior performance in solving classic optimization functions, high-dimensional functions, high-dimensional cases, and faster convergence to optimal solutions [29-36].

In the next subsection, we will explain the Whale Optimization Algorithm (WOA) in detail. This section will cover basic principles and operational mechanics in the context of optimization problems.

3.1 Whale optimization algorithm (WOA)

The Whale Optimization Algorithm (WOA) is a metaheuristic optimization method that draws inspiration from the foraging habits of whales. It is widely utilized for global optimization challenges in various domains. Unlike other hunting strategies, WOA stands out by employing either random or the best agent within the search space to pursue prey. The algorithm replicates the bubble-net attachment mechanisms of humpback whales, who use distinctive spirals to encircle and capture groups of krill or small fish near the water's surface. This involves the formation of spiral bubbles around the prey as the whales move up and down in the water as shown in Fig.1.

The WOA mimics whale behavior by employing three strategies that imitate the actions of humpback whales during the phases of searching for prey (exploration), encircling prey, and bubble-net foraging (exploitation). The subsequent sections will provide a theoretical framework and elucidate the mathematical formulation.

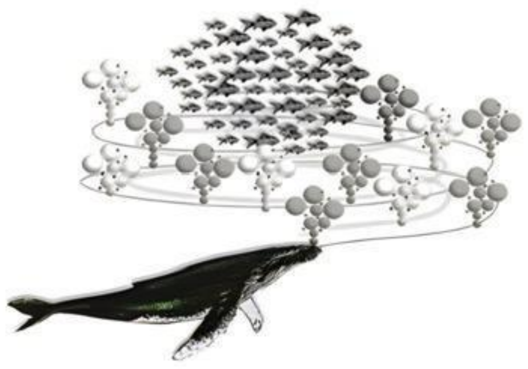
Figure. 1 The graph showing the spiral bubble-net attacking strategy [37]

Table 2. Notation list

Variables	Meaning
H	current iteration
$\vec{d}^*$	location of the best solution
$\vec{d}$	location vector
$\vec{z}$ and $\vec{y}$	coefficient vectors
K	random vector
W	vector
$\overrightarrow{d\ rand}$	location selected

Table 2 displays a notation list. It explains the meaning of the variables in mathematical equations.

3.1.1. Prey encirclement

The whale algorithm begins this phase by selecting an initial optimal search agent. The prevailing assumption is that the current solutions are optimal and that they are located in close proximity to the prey. Consequently, the other agents modify their positions toward the most optimal search agent. This is represented as follows:

$$\vec{s} = \vec{y}.\vec{d}^*(t) - d(h) \qquad (1)$$

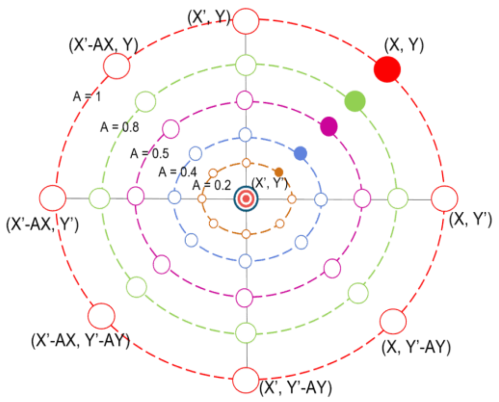

Figure. 2 WOA shrinking encircling mechanism [38]

Received: March 5, 2024. Revised: April 6, 2024.

$$\vec{d} = (h+1)\vec{d}^*(h) - \vec{z}.\vec{s} \tag{2}$$

Where h represents the current iteration $\vec{z}$ and $\vec{y}$ are coefficient vectors. The vector $\vec{d}^*$ represents the location of the best solution that has been achieved thus far, while $\vec{d}$ represents the location vector. If a more optimal solution is found, the variable d^* should be updated by iterative processes. The calculation for vectors $\vec{z}$ and $\vec{y}$ is as follows:

$$\vec{z} = 2\,\vec{w}.\vec{k} - w \tag{3}$$

$$\vec{y} = 2.\vec{k} \tag{4}$$

K is a random vector ranging from 0 to 1, and the vector w is linearly decreased from 2 to 0 over the iterations. This model replicates the prey's surroundings and enables each agent to adjust its position within the current optimal solution area. As illustrated in Fig. 2. The search can extend deeper into the n-dimensional space, facilitating navigation around the hypercube for agents near the optimal solution.

3.1.2. Exploitation phase

This phase is also known as the bubble-net attack, and it employs the following two strategies:

Shrinking encircling mechanism: In this phase, the value of $\vec{w}$ in Eq. (3) is reduced, reducing the fluctuation range of $\vec{z}$ by $\vec{w}$. This indicates that a is arbitrarily located in $[\overline{-w}, \vec{w}]$. Where a decreases from 2 to 0 during the optimization process.

Because of The randomization of $\vec{z}$ in $[-1,1]$, the search agent's new position, can be found anywhere between the agent's previous location and the current optimal location. In a 2-D space, Fig. 2 presents the various locations based on the shrinking encircling mechanism.

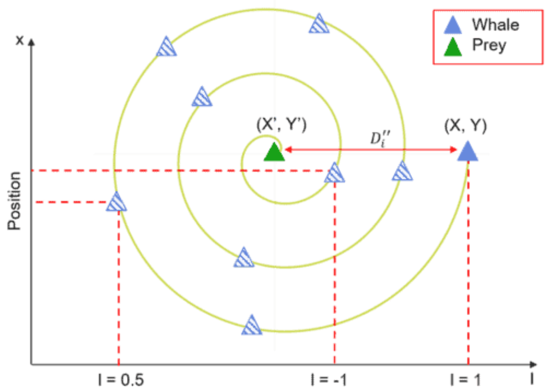

Figure. 3 WOA spiral updates its position [13]

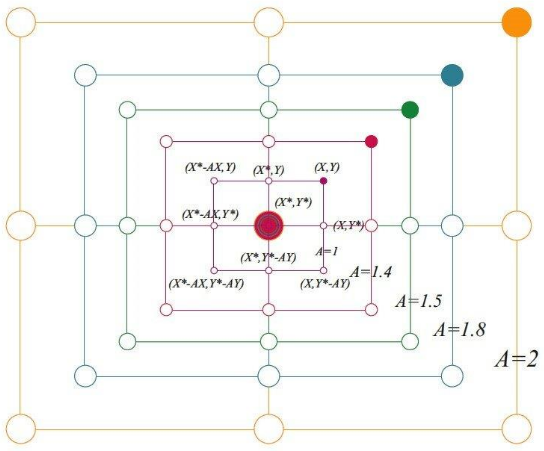

Figure. 4 WOA exploration mechanism [38]

Spiral position updating: During this stage, the computation involves determining the distance separating the whale from its prey and establishing a spiral equation based on their respective positions, mirroring the feeding behavior exhibited by the whale as illustrated in Fig. 3.

Humpback whales move the form of the spiral. This may be represented as follows:

$$\vec{d}(\mathrm{h}+1)\vec{L'}\,.\,e^{bl}.\cos(2\Pi\iota) + \vec{d}^*(h) \tag{5}$$

$$\vec{L'} = \left|\vec{\mathrm{X}}^*(\mathrm{h}) - \vec{\mathrm{d}}(h)\right| \tag{6}$$

The distance between the whale and its prey is represented by Eq. (6) (the best solution so far), where b is a constant for determining the logarithmic spiral shape and ι is a random number between $[-1,1]$. A spiraling pattern of shrinking circles follows the whale as it moves toward its prey. As a result, a 50% probability of switching between modes is employed to update the whale's next position, as shown below:

$$\vec{d}(h+1) = \begin{cases} \vec{d^*}(t) - \vec{z}.\vec{L} & , if\ p \geq 0.5 \\ \vec{l'}\,.\,e^{bl}.\cos(2\Pi\iota) + \vec{\mathrm{d}}^*(t), & if\ p \geq 0.5 \end{cases} \tag{7}$$

Where p in $[0,1]$ is a random number.

3.1.3. Exploration phase

WOA simulates global optimization at this phase. As shown in Fig. 4, whales seek prey at random based on their relative positions to one another.

The vector (z) is randomly assigned a value within the range of -1 to 1 in order to compel the search agent to distance itself from the reference whale. The vector (z) ⃗ must have a magnitude greater than 1 or less than -1. Here, a randomly selected agent enables the WOA to perform a worldwide search and update the agent's position. The exploration process is depicted as follows:

$$\vec{L} = \left|\vec{y}.\overrightarrow{Xrand} - \vec{d}\right| \tag{8}$$

$$\vec{d}(h+1) = \overrightarrow{drand} - \vec{z}.\vec{L} \tag{9}$$

Where $\overrightarrow{d\,rand}$ is a location selected at random for a whale from the current population. The algorithm (1) illustrates the pseudo-code for the phases of the WOA.

3.2 CIC- MalMem-2022 dataset

Obfuscated malware is a form of malicious software that employs concealment methods to avoid being detected and eliminated. The main goal of the obfuscated malware dataset is to evaluate the effectiveness of malware detection methods. The Canadian Institute for Cybersecurity has launched CIC-Malmem-2022, an academic dataset designed for studies on malware classification, with a specific focus on obfuscated malware. This dataset aims to simulate real-world conditions effectively by including common malware samples, encompassing Spyware, Ransomware, and Trojan Horse malware. Fig. 5 illustrates the categories of memory dumps.

To expand the dataset, an automated process was implemented that included the execution of 2,916 malware samples representing three different categories: Trojan horse, ransomware and spyware, within a virtual machine (VM). As it is important to disrupt benign processes when creating malicious memory dumps, various applications within the Windows VM were launched simultaneously with the execution of malware samples. Each malware sample execution provided 10 memory dumps recorded at 15-second intervals, resulting in a total of 29,298 malicious memory dumps to comprehensively capture the potential behavior of malware [28].

Initialize the whales' population $X_{i(i=1,2,3,\dots,n)}$.
Compute the fitness of each whale.
Set X^*as the best whale.

Algorithm1: The WOA algorithm [13]

```
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, $A$, $C$,$l$ and $p$.
        If ($p$ < 0.5) then
          If (|$A$|< 1) then
              The whale position is updating by
              the Eq. (1).
          Else
              If (|$A$| ≥1) then
                  Select the random whale $X_{rand}$
                  The whale position is updating by
                  the Eq. (9).
              End If
          End If
        Else
          If ($p \geq 0.5$) then
              Modify the whale position by the
              Eq. (5).
          End If
        End If
    End For
    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 While
```

Benign dumps were generated by observing typical user behavior through the activation of various applications on the computer. Oversampling was then performed using the SMOTE (Synthetic Minority Over-sampling Technique) algorithm to equalize the data set. In contrast to conventional oversampling methods, SMOTE generates synthetic values that differ only minimally from the actual values. This methodological choice helps to create a more balanced and representative dataset for subsequent analysis and classification efforts [28].

The dataset is evenly divided, comprising 50% malicious memory dumps and 50% benign memory dumps. Fig. 6 shows the overall number of malware families in each malware category.

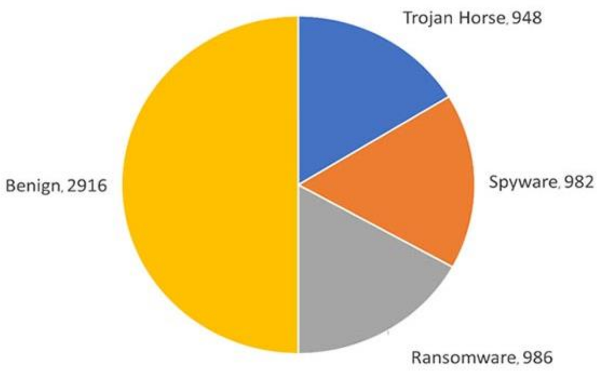

Figure. 5 Memory Dump Categories [28]

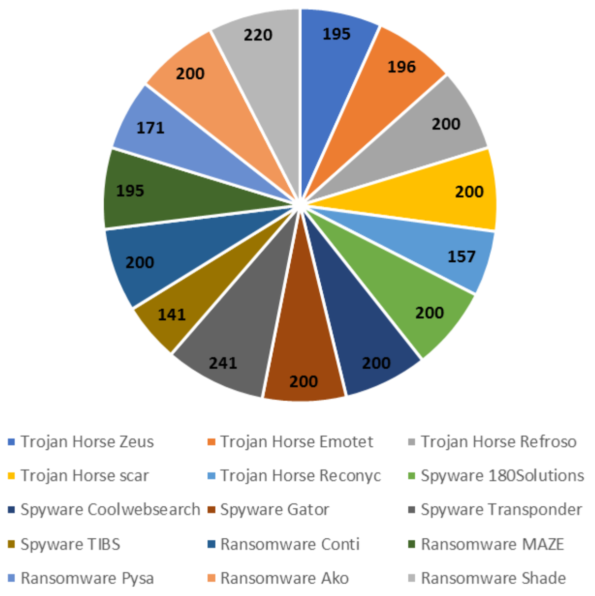

Figure. 6 Malware Breakdown

3.2.1. Preprocessing dataset

The dataset was pre-processed to maintain consistency and improve the optimization algorithms' performance, as detailed in the following subsections. **Normalization:** Normalization is essential for ensuring the numerical features within the CIC-Malmem-2022 dataset are on the same scale. This step prevents features with larger values from dominating the analysis, which is critical in the context of classifying data. Min-Max scaling was applied to scale the values to a range between 0 and 1, promoting consistent and unbiased analysis. To achieve this, Min-Max scaling was applied, utilizing the following formula:

$$Normalize\ Values\ \frac{Original\ Value - Min(A)}{Max(A) - Min(A)} \quad (10)$$

Where:

- Original Value: Depicts the initial numerical values of the dataset.
- $Min(A)$: Denotes the lowest value of feature A in the dataset.
- $Max(A)$: Denotes the highest value of feature A in the dataset.

Rounding Digits: rounding the digits to a specific attribute within the CIC-Malmem-2022 dataset to enhance generalization and prevent overfitting. Numbers have been rounded to four decimal places to ensure that any numerical values are displayed with a high degree of accuracy.

3.2.2. Model training and testing

The dataset has been divided into two separate subsets: 80% of the data is assigned for training the model, while the remaining 20% is set aside for testing the model's performance. This partitioning strategy aligns with common practices observed in other studies [39, 40]. In the training phase, the KNN classifier undergoes training using the designated training data, with emphasis placed on the subset of optimized features.

4. The proposed method

This section outlines the steps involved in using EWOA to optimize feature selection, as shown in Fig. 7.

4.1 Enhanced whale optimization algorithm (EWOA)

In this section, an enhanced version of WOA is presented. Two enhancements have been added to the EWOA. Firstly, the initialization phase in EWOA has been replaced by OBL to improve the quality of selecting the position of the whale (agent) in the initialization phase. Secondly, a new search strategy, namely Search Strategy (SS), has been proposed to enhance the search mechanism of EWOA in both the exploration and exploitation phases, as shown in Fig. 8.

EWOA is designed to address challenges and limitations that were observed in the original WOA, thus providing a more resilient and adaptable optimization tool.

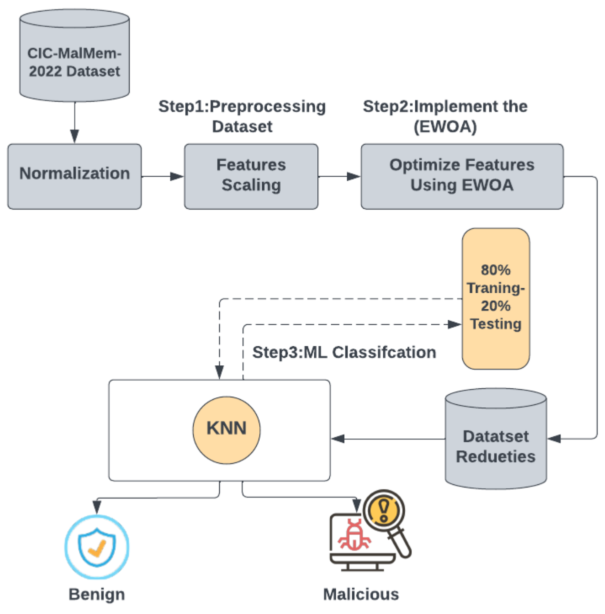

Figure. 7 EWOA for feature selection.

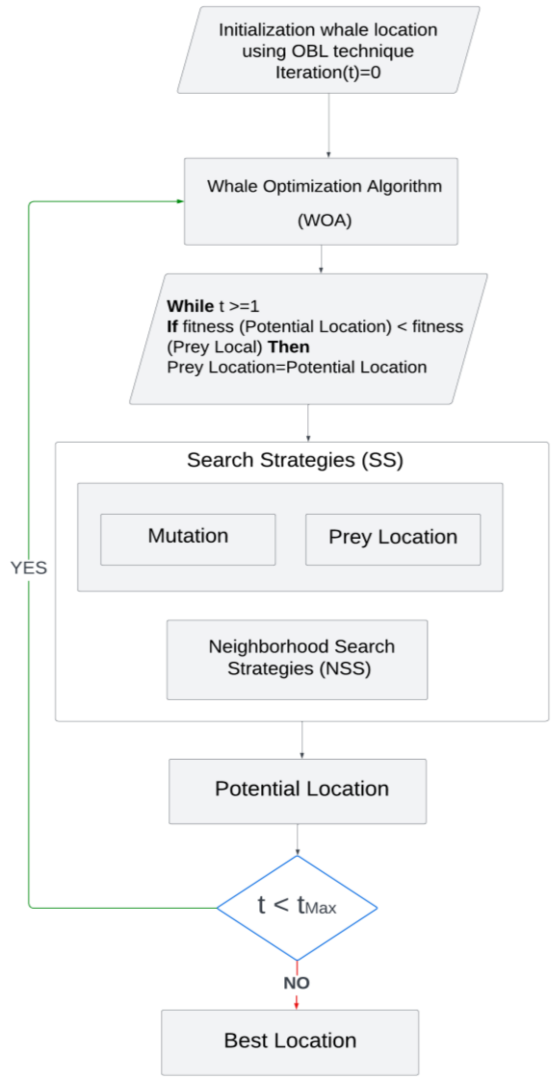
Initialization whale location using OBL technique Iteration(t)=0
Whale Optimization Algorithm (WOA)
While t >=1
If fitness (Potential Location) < fitness (Prey Local) Then
Prey Location=Potential Location
Search Strategies (SS)
Mutation
Prey Location
Neighborhood Search Strategies (NSS)
Potential Location
YES
t < tMax
NO
Best Location

Figure. 8 The framework of the proposed OBL and SS

WOA is a robust and effective optimization technique. However, as per the NFL theorem, no algorithm can completely solve all optimization problems [41]. Enhancements have been made to the WOA algorithm to address its limitations and enhance its feature selection optimization capabilities. This enhancement aims to improve the algorithm's abilities in global and local searches by incorporating the suggested SS. The subsequent steps elucidate the proposed EWOA algorithm.

Step 1: Initialize the population, represented as X, with a size of N using a random function. Each whale in the population represents a different feature subset. Feature subsets are represented as binary vectors, where each element corresponds to the presence or absence of a feature.

Step 2: Utilize the OBL technique to create solutions that are opposite in nature, then choose the most suitable N solutions.

Step 3: Apply the Enhanced Whale Optimization Algorithm (EWOA) to adjust the position of each member in the population. Determine the optimal prey location based on the highest fitness value. Setting up EWOA Parameters [42]. The EWOA algorithm requires setting parameters in which the number of whales (20), maximum iterations (30), and exploration/exploitation factors. Executing the EWOA Algorithm: EWOA iteratively updates the positions of whales (feature subsets) in the search space. Whales mimic the behavior of humpback whales, moving towards the optimal search agent to improve the fitness of their feature subsets. Eq. (1) and Eq. (2) govern the updating process, guiding the exploration and exploitation of the search space.

Step 4: Implement a mutation strategy to enhance prey detection. If the new location is more suitable than the current location, classify the new location as a potential prey location. Execute the NSS strategy to improve its positioning. Ultimately, designate the optimal location as the potential prey locations.

Step 5: Iterate until the termination condition is satisfied.

The following subsections will explain in detail the techniques used to enhance an EWOA algorithm.

- **Opposite-Based Learning**

The OBL technique, created by [35] OBL is a method that encompasses a machine learning approach. This technique aims to improve the efficiency of metaheuristic optimization algorithms. This strategy entails choosing a more efficient solution from the existing individuals, usually randomly initialized by the optimization algorithm, along with its corresponding opposite solution. Each solution is assigned fitness values, and the one with the highest value is selected to progress to the next iteration. Research has shown that OBL greatly improves the likelihood of reaching the best global solution for a specific objective function, thus boosting the effectiveness of optimization algorithms [43]. The OBL technique can be mathematically formulated as follows.

$Let\ d = (d1, d2, \dots, dD)$ represents a location in the current population, while D denotes the dimensionality of the problem space. and $x \in [ai, bi]$, $i = 1, 2, \dots, D$. the opposition point $\check{d} = (\check{d}_1, \check{d}_2, \dots, \check{d}_D)$ is thus defined as the following equation:

$$\check{d}_i = a_i + b_i - d_i \tag{11}$$

Optimization algorithms involve exploration (diversification) and exploitation (intensification) phases. This section outlines the proposed Search

Strategy (SS) designed to enhance the global and local search mechanisms in the EWOA algorithm. These strategies partially reduce the risk of getting trapped in local optima. The proposed SS consists of:

- **Mutation**

Mutation in the EWOA algorithm aims to increase diversity in the sampled population. Mutation operators are used to avoid the chromosomes in a population from reaching a local optimum by ensuring they do not become too similar to each other. Various mutation categories exist, based on the technique used. The enhanced algorithm utilizes bit string mutation, randomly altering features by flipping them at random positions. The mathematical representation of bit string mutation for the solution $d = (d1, d2, \dots, dL)$ is as follows:

$$M\,(m) = |1 - D\,(m)| \tag{12}$$

Where M represents the solution resulting from executing bit string mutation, $m = 1, 2, \dots, L$ it is a matrix of randomly selected positions (features) to be flipped in solution d. In solution d, specifically, the third and sixth features are undergoing a flipped, as shown in Fig. 9.

The mutation size was randomly selected between 10% and 50% in the exploration phase and between 1% and 9% in the exploitation phase through various trial and error experiments. We introduced a technique in the EWOA to determine the degree of changes in features. During the exploration phase, selected features from the current optimal position are flipped to improve the efficiency of the global search. In the exploitation phase, a small number of features are altered to improve local search abilities. The mutation size is defined as follows:

$$Mutation_{size} = \begin{cases} Number\ of\ Features * \dfrac{10 * rand\,[1,5]}{100}, & if\ |h| \le \dfrac{H}{2} \\ Number\ of\ Features * \dfrac{rand\,[1.9]}{100}, & if\ |h| > \dfrac{H}{2} \end{cases} \tag{13}$$

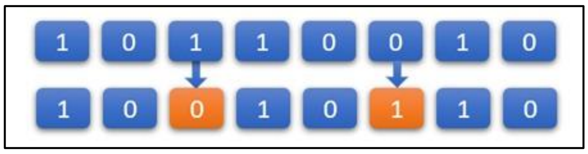

Figure. 9 An instance of bit string mutation involving the inversion of the 3rd and 6th features

Where H represents the current iteration and H Maximum iteration.

- **Neighborhood search strategies (NSS)**

Das et al. (2009) implemented neighbor search in differential evolution (DE) to achieve a balance between exploration and exploitation phases. The primary objective of neighbor search is to explore a limited area surrounding the current optimal solution, rather than the entire population. This study presents the Neighborhood Search Strategy (NSS) technique. The implementation of NSS depends on improvements made to the best solution produced by the mutation strategy. NSS is utilized when the existing optimal solution (prey location) is altered as a result of mutation. The fitness value is recalculated following the application of each mutation to the current optimal position [44, 45].

The process includes assessing the suitability of the recently mutated position. When the new position's fitness exceeds that of the current position, the current optimal solution is replaced by the mutated solution, and the neighborhood search is performed. NSS specifically examines two neighboring strategies for toggling features. The forward-switching technique involves altering the right feature and evaluating the fitness values of both solutions (the best solution and the current switched solution). The backward-switching technique involves using the same method to mutate the left feature. Ultimately, this leads to two solutions, with the optimal value being recognized as the most favorable solution. The NSS is structured as a ring, where the final feature is linked to the initial feature to create neighboring connections on both ends. Fig. 10 displays the ring NSS strategy.

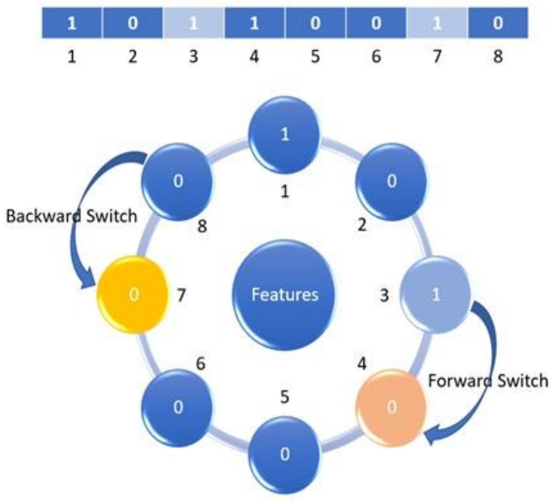

Figure. 10 An illustration of Neighborhood Search Strategies (NSS)

If the new position generated by mutation consistently maintains the best fitness value for consecutive iterations, it is referred to as the NSS strategy within the SS approach.

WOA was initially created for continuous solution search spaces. Modifications are required to be in line with binary feature selection. Each whale's position is transformed into binary solutions using the following method.

$$x_{i,j} = \begin{cases} 1 \ \ if \ \dfrac{1}{1+e^{-xij}} \geq 0.5 \\ 0 \ \ otherwise, \end{cases} \qquad (14)$$

Consequently, only the features that match the ones in the dataset are considered relevant features, whereas the features that match the zeros are ignored.

Selecting the optimized feature subset: After the EWOA algorithm converges or reaches the maximum number of iterations, the subset of features that produces the highest classification accuracy is selected as the optimized set.

EWOA is utilized dynamically to identify the optimal feature subset that enhances classification accuracy. Whales in EWOA exhibit dynamic movement within the search space, starting from the optimal search agent. Subsequently, they attempt to adjust their positions towards the most efficient search agent, as depicted in Eq. (1) and Eq. (2).

KNN classifies unknown examples based on the majority class among its KNN in the feature space [38]. Specifically, we use KNN to classify samples into malware or benign categories using a selected feature subset. The classification strategy guides the feature selection method based on optimized components and a chosen feature set.

5. Experimental results and discussion

This section will detail the experimental analysis and outcomes of the proposed approach. Comparative evaluations among EWOA, WOA, GOA, GA, PSO, ALO, BOA, and SMA will be conducted, considering metrics such as the average number of features, efficiency, fitness value, accuracy, and statistical tests. The assessment is based on 30 iterations to ensure the reliability and consistency of the obtained results.

According to the results, the proposed method is accurate and efficient. In Table 3, the average number of features is calculated using different optimization algorithms, and the results indicate that EWOA is the most efficient algorithm according to the lowest number of selected features.

The average number of features is essential in determining their effectiveness. The EWOA algorithm has the lowest average number of features at 3.9667 out of 55 total features, suggesting it utilizes fewer features compared to the other algorithms. As a result, EWOA is superior at selecting a relevant subset of features, which may enhance the model by reducing the problem of dimensionality in the most efficient manner. Conversely, GOA, GA, PSO, BOA, and SMA show higher average numbers of features, ranging from 19.3667 to 24.9, indicating a broader inclusion of features in their solutions. WOA and ALO present intermediate values at 7.2333 and 12.7, respectively. Table 4. Displays the average accuracy outcomes across different optimization algorithms.

When evaluating the performance of various algorithms based on average accuracy, the optimization algorithms show high average accuracy values. Notably, EWOA stands out with the highest average accuracy of 0.99987, followed by PSO, GA, GOA, and BOA, all of which indicate accuracy values exceeding 0.9995. SMA and ALO also present impressive accuracy, ranging from 0.9993 to 0.9996. In addition, the WOA algorithm has an average accuracy of 0.99904.

Table 3. The average number of feature outcomes.

Algorithm	**Average Number of Features**
EWOA	3.9667
GOA	24.0667
GA	22.2667
PSO	19.3667
ALO	12.7
WOA	7.2333
BOA	24.6
SMA	24.9

Table 4. The average results of accuracy.

Algorithm	**Average Accuracy**
EWOA	0.99987
GOA	0.99956
GA	0.9996
PSO	0.99969
ALO	0.99927
WOA	0.99904
BOA	0.99956
SMA	0.99952

Table 5. Shows the average time results.

Algorithm	Average Time (Seconds)
EWOA	43.1877
GOA	80.2727
GA	52.1437
PSO	71.7279
ALO	45.1597
WOA	27.0792
BOA	333.6596
SMA	141.0108

Table 6. The average fitness values results.

Algorithm	Average fitness
EWOA	0.00084511
GOA	0.004815
GA	0.0044483
PSO	0.0038253
ALO	0.0030356
WOA	0.0022613
BOA	0.004912
SMA	0.005006

This investigation achieved a noteworthy elevation in precision, demonstrating a remarkable accuracy of 99.987% through the application of the KNN classifier. This performance surpasses the accuracy in the comparative study by [28]. Wherein a classification accuracy of 0.95 was attained utilizing the KNN classifier. The considerable improvement in accuracy observed in our study indicates the substantive progress realized in the proposed model. Table 5. Shows the average computational times of the optimization algorithms, providing insight into their efficiency.

Both EWOA and WOA algorithms exhibit significantly lower average execution times, indicating a higher degree of computational efficiency. Conversely, the BOA algorithm stands out with a significantly longer computational time, indicating a relatively slower optimization process. The GOA, GA, ALO, PSO, and SMA algorithms demonstrate moderately comparable computational durations. In terms of preference, EWOA and WOA showcase faster convergence, while BOA requires considerably more computational resources. Table 6. Shows average fitness values across optimization algorithms that represent valuable insights into their relative effectiveness in achieving optimal solutions.

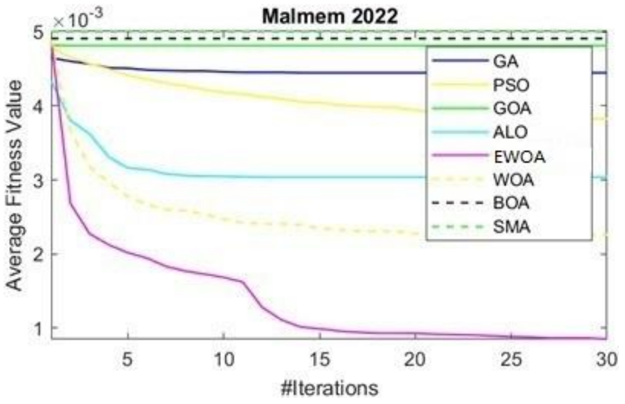

Figure. 11 The average fitness values of EWOA across different optimization algorithms.

Table 7. Shows the p-value results.

Algorithm	p-value
GOA	2.7495e-11
GA	2.746e-11
PSO	2.7322e-11
ALO	2.7547e-11
WOA	3.1995e-11
BOA	2.7478e-11
SMA	2.7478e-11

It offers valuable information on how various optimization algorithms compare in terms of achieving optimal solutions when looking at average fitness values. A lower average fitness value indicates greater algorithm efficiency in minimizing the objective function. The EWOA algorithm is the most effective among the algorithms in this context, boasting an impressive average fitness of 0.00084511. This indicates that EWOA is a highly promising solution for optimization tasks where the objective is to find optimal solutions with minimal fitness values.

Similarly, the WOA and ALO algorithms show impressive performance, showcasing low average fitness values, respectively, at 0.0022613 and 0.0030356. These findings indicate the efficacy of WOA and ALO in converging towards solutions with minimal fitness values, reflecting their ability to navigate the solution space effectively. The PSO algorithm also demonstrated competitive performance with an average fitness of 0.0038253. Conversely, the SMA shows the highest average fitness value at 0.005006, indicating a relatively less efficient exploration of the solution. GA, GOA, and BOA show intermediate fitness values of 0.0044483, 0.004815, and 0.004912, respectively. These algorithms show acceptable efficiency. Fig. 11

shows the average fitness values of EWOA across different optimization algorithms.

The graphical representation illustrates that the optimal solution was attained at iteration 15, marking the midpoint of the total 30 iterations. Subsequent iterations displayed a plateau in the improvement of solutions, indicating a stabilization in the optimization process. This observation implies that optimal solutions were efficiently achieved with a minimal number of iterations. The convergence of solution improvement after the 15th iteration supports this, reinforcing the notion that the algorithm reached an effective and optimal state early in the iterative process. Table 7 displays the P-values across different optimization algorithms, providing insight into evaluating the significance of the improvement.

The P-values from different optimization algorithms: GOA, GA, PSO, ALO, WOA, BOA, and SMA, indicate low values ranging from 2.7322e-11 to 3.1995e-11. In statistical terms, a lower P-value is indicative of greater statistical significance. The significance level, often set at 0.05, indicates the threshold below which results are considered statistically significant. As shown, all the P-values fall significantly below this threshold, indicating that the improvements achieved by EWOA to these optimization algorithms are highly statistically significant.

EWOA emerges as a robust and effective optimization algorithm for malware detection, exhibiting superiority in critical evaluation metrics including feature selection, computational efficiency, fitness value, and overall model accuracy. These compelling results underscore the capacity of EWOA to notably augment the effectiveness of malware detection systems, offering a promising alternative to established optimization algorithms.

Conclusion

This paper introduces a model that combines malware analysis, feature selection, and machine learning for malware detection. We propose a novel malware detection system based on the Enhanced Whale Optimization Algorithm (EWOA). We utilize EWOA for feature reduction to effectively manage the complexity associated with high-dimensional data in machine learning.

By combining mutation strategies and neighborhood search strategies, EWOA improves its performance in avoiding local optima. In addition, EWOA utilizes the Opposition-Based Learning (OBL) technique to enhance the diversity of its populations.

Our findings indicate that EWOA surpasses all other optimization algorithms in features selection for detecting malware. We achieved an optimal accuracy using an enhanced WOA based on KNN, surpassing the previously reported highest accuracy. Moreover, EWOA reduced the number of features from 55 to an average of 3.97 features in the CIC-Malmem-2022 dataset. It is noteworthy that, to the best of our knowledge, this research marks the first instance of reducing the number of features in the CIC-Malmem-2022, thereby enhancing overall classification performance.

Future applications of this model may include the detection of polymorphic and metamorphic malware. It is also of great interest to investigate the efficacy of deep learning approaches such as recurrent neural networks, long- and short-term memory, and others in detecting malware. The proposed search strategies can also be applied to other optimization algorithms to enhance their efficiency, introducing new avenues for refining, and advancing existing optimization methodologies. Moreover, the adaptability of this model to alternative datasets provides an exciting opportunity to evaluate its efficacy in varied contexts, thereby expanding its scope and utility.

Conflicts of Interest

The authors declare no conflict of interest.

Author Contributions

Conceptualization, D.I. and M.A.; methodology, M.A; software, R.S.; validation, M.A, D.I., and R.S.; formal analysis, R.S.; investigation, M.A.; resources, D.I; data curation, M.A.; writing—original draft preparation, M.A; writing—review and editing, D.I.; visualization, M.A; supervision, R.S.; funding acquisition, R.S.; project administration, D.I. and R.S.;

References

[1] F. T. Ngo, A. Agarwal, R. Govindu, and C. MacDonald, "Malicious software threats", *The Palgrave Handbook of International Cybercrime and Cyberdeviance*, pp. 793-813, 2020.

[2] A. Firdaus, N. B. Anuar, and S. A Razak. “Survey of malware detection techniques”, *Journal of Network and Computer Applications*, vol. 60, no. 19-31, 2017.

[3] Ö. A. Aslan and R. Samet, "A comprehensive review on malware detection approaches", *IEEE Access*, Vol. 8, pp. 6249-6271, 2020.

[4] M. Jerbi, Z. C. Dagdia, S. Bechikh, and L. B. Said, "Immune-Based System to Enhance Malware Detection", In: *Proc. of IEEE 2023 Congress on Evolutionary Computation*, 2023.

[5] J. Singh and J. Singh, "A survey on machine learning-based malware detection in executable files", *Journal of Systems Architecture,* Vol. 112, pp. 101861, 2021.

[6] R. Sihwail, K. Omar, and K. A. Z. Arifin, "An Effective Memory Analysis for Malware Detection and Classification", *Computers, Materials & Continua,* Vol. 67, No. 2, 2021.

[7] Y. Li, K. Xiong, T. Chin, and C. Hu, "A machine learning framework for domain generation algorithm-based malware detection", *IEEE Access,* Vol. 7, pp. 32765-32782, 2019.

[8] C. Opris, "Machine learning techniques for the analysis and detection of malicious software", *Ph.D. dissertation, Cybersecurity, Technical University,* Cluj-Napoca, 2021.

[9] F. A. Aboaoja, A. Zainal, F. A. Ghaleb, B. A. S. Al-rimy, T. A. E. Eisa, and A. A. H. Elnour, "Malware detection issues, challenges, and future directions: A survey", *Applied Sciences*, Vol. 12, No. 17, p. 8482, 2022.

[10] W. Jia, M. Sun, J. Lian, and S. Hou, "Feature dimensionality reduction: a review," *Complex & Intelligent Systems,* Vol. 8, No. 3, pp. 2663-2693, 2022.

[11] R. Zebari, A. Abdulazeez, D. Zeebaree, D. Zebari, and J. Saeed, "A comprehensive review of dimensionality reduction techniques for feature selection and feature extraction", *Journal of Applied Science and Technology Trends,* Vol. 1, No. 2, pp. 56-70, 2020.

[12] J. Abawajy, A. Darem, and A. A. Alhashmi, "Feature subset selection for malware detection in smart IoT platforms", *Sensors*, Vol. 21, No. 4, p. 1374, 2021.

[13] S. Mirjalili and A. Lewis, "The whale optimization algorithm", *Advances in engineering software,* Vol. 95, pp. 51-67, 2016.

[14] N. Rana, M. S. A. Latiff, S. i. M. Abdulhamid, and H. Chiroma, "Whale optimization algorithm: a systematic review of contemporary applications, modifications and developments", *Neural Computing and Applications,* Vol. 32, pp. 16245-16277, 2020.

[15] I. Aljarah, H. Faris, and S. Mirjalili, "Optimizing connection weights in neural networks using the whale optimization algorithm", *Soft Computing,* Vol. 22, pp. 1-15, 2018.

[16] Y. Shen, C. Zhang, F. S. Gharehchopogh, and S. Mirjalili, "An improved whale optimization algorithm based on multi-population evolution for global optimization and engineering design problems", *Expert Systems with Applications,* Vol. 215, pp. 119269, 2023.

[17] D. Smith, S. Khorsandroo, and K. Roy, "Leveraging Feature Selection to Improve the Accuracy for Malware Detection", *PREPRINT,* 2023.

[18] N. A. Alawad, B. H. Abed-alguni, M. A. Al-Betar, and A. Jaradat, "Binary improved white shark algorithm for intrusion detection systems", *Neural Computing and Applications,* pp. 1-25, 2023.

[19] R. Sihwail, K. Omar, K. A. Z. Ariffin, and M. Tubishat, "Improved harris hawks optimization using elite opposition-based learning and novel search mechanism for feature selection", *IEEE Access,* Vol. 8, pp. 121127-121145, 2020.

[20] A. Rouhi and H. Nezamabadi-Pour, "Feature selection in high-dimensional data", *Optimization, Learning, and Control for Interdependent Complex Networks,* pp. 85-128, 2020.

[21] S. Chakraborty, A. K. Saha, R. Chakraborty, and M. Saha, "An enhanced whale optimization algorithm for large scale optimization problems", *Knowledge-Based Systems,* Vol. 233, pp. 107543, 2021.

[22] S. Chakraborty, A. K. Saha, S. Sharma, S. Mirjalili, and R. Chakraborty, "A novel enhanced whale optimization algorithm for global optimization", *Computers & Industrial Engineering,* Vol. 153, pp. 107086, 2021.

[23] M. A. Talukder *et al.,* "A dependable hybrid machine learning model for network intrusion detection", *Journal of Information Security and Applications, V*ol. 72, pp. 103405, 2023.

[24] D. Smith, S. Khorsandroo, and K. Roy, "Supervised and unsupervised learning techniques utilizing malware datasets", In: *Proc. of 2023 IEEE 2nd International Conference on AI in Cybersecurity (ICAIC),* pp. 1-7, 2023.

[25] S. S. Shafin, G. Karmakar, and I. Mareels, "Obfuscated Memory Malware Detection in Resource-Constrained IoT Devices for Smart City Applications", *Sensors,* Vol. 23, No. 11, pp. 5348, 2023.

[26] M. Dener, G. Ok, and A. Orman, "Malware detection using memory analysis data in big data environment", *Applied Sciences,* Vol. 12, No. 17, pp. 8604, 2022.

[27] J. Luhr and H. Hallqvist, "Fast Classification of Obfuscated Malware with an Artificial Neural Network", *Dissertation, Computer Science, KTH, School of Electrical Engineering and*

Computer Science (EECS), Stockholm, Sweden, 2022.

[28] T. Carrier, "Detecting Obfuscated Malware Using Memory Feature Engineering", *Master Thesis, Computer Science*, New Brunswick, New Jersey, United States, 2021.

[29] P. D. Kusuma and A. Dinimaharawati, "Extended stochastic coati optimizer", *International Journal of Intelligent Engineering and Systems,* Vol. 16, No. 3, pp. 482-494, 2023., doi: 10.22266/ijies2023.0630.38.

[30] P. D. Kusuma and A. Dinimaharawati, "Swarm Bipolar Algorithm: A Metaheuristic Based on Polarization of Two Equal Size Sub Swarms", *International Journal of Intelligent Engineering and Systems,* Vol. 17, No. 2, 2024, doi: 10.22266/ijies2024.0430.31.

[31] P. D. Kusuma and M. Kallista, "Swarm Space Hopping Algorithm: A Swarm-based Stochastic Optimizer Enriched with Half Space Hopping Search", *International Journal of Intelligent Engineering and Systems*, Vol. 17, No. 2, 2024, doi: 10.22266/ijies2024.0430.54.

[32] P. D. Kusuma and M. Kallista, "Migration-Crossover Algorithm: A Swarm-based Metaheuristic Enriched with Crossover Technique and Unbalanced Neighbourhood Search", *International Journal of Intelligent Engineering and Systems,* Vol. 17, No. 1, 2024, doi: 10.22266/ijies2024.0229.59.

[33] P. D. Kusuma and A. Novianty, "Total Interaction Algorithm: A Metaheuristic in which Each Agent Interacts with All Other Agents", *International Journal of Intelligent Engineering & Systems,* Vol. 16, No. 1, 2023, doi: 10.22266/ijies2023.0228.20.

[34] P. D. Kusuma and A. Dinimaharawati, "Four Directed Search Algorithm: A New Optimization Method and Its Hyper Strategy Investigation", *International Journal of Intelligent Engineering and Systems,* Vol. 16, No. 5, 2023, doi: 10.22266/ijies2023.1031.51.

[35] P. D. Kusuma and F. C. Hasibuan, "Attack-Leave Optimizer: A New Metaheuristic that Focuses on The Guided Search and Performs Random Search as Alternative", *International Journal of Intelligent Engineering and Systems,* Vol. 16, No. 3, 2023, doi: 10.22266/ijies2023.0630.19.

[36] P. D. Kusuma and A. L. Prasasti, "Walk-Spread Algorithm: A Fast and Superior Stochastic Optimization", *International Journal of Intelligent Engineering & Systems,* Vol. 16, No. 5, 2023, doi: 10.22266/ijies2023.1031.24.

[37] M. H. Nadimi-Shahraki, H. Zamani, Z. Asghari Varzaneh, and S. Mirjalili, "A Systematic Review of the Whale Optimization Algorithm: Theoretical Foundation, Improvements, and Hybridizations", *Archives of Computational Methods in Engineering,* pp. 1-47, 2023.

[38] M. Sharawi, H. M. Zawbaa, and E. Emary, "Feature selection approach based on whale optimization algorithm", In: *Proc. of 2017 Ninth international conference on advanced computational intelligence (ICACI),* pp. 163-168, 2017.

[39] S. Gamage and J. Samarabandu, "Deep learning methods in network intrusion detection: A survey and an objective comparison", *Journal of Network and Computer Applications,* Vol. 169, pp. 102767, 2020.

[40] M. Gopinath and S. C. Sethuraman, "A comprehensive survey on deep learning based malware detection techniques", *Computer Science Review,* Vol. 47, pp. 100529, 2023.

[41] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization", *IEEE Transactions on Evolutionary Computation,* Vol. 1, No. 1, pp. 67-82, 1997.

[42] D. Cao, Y. Xu, Z. Yang, H. Dong, and X. Li, "An enhanced whale optimization algorithm with improved dynamic opposite learning and adaptive inertia weight strategy", *Complex & Intelligent Systems,* Vol. 9, No. 1, pp. 767-795, 2023.

[43] H. R. Tizhoosh, "Opposition-based learning: a new scheme for machine intelligence", In: *Proc. of International conference on computational intelligence for modelling, control and automation and international conference on intelligent agents, web technologies and internet commerce (CIMCA-IAWTIC'06),* Vol. 1, pp. 695-701, 2005.

[44] S. Das, A. Abraham, U. K. Chakraborty, and A. Konar, "Differential evolution using a neighborhood-based mutation operator", *IEEE Transactions on Evolutionary Computation,* Vol. 13, No. 3, pp. 526-553, 2009.

[45] R. Sihwail, O. S. Solaiman, and K. A. Z. Ariffin, "New robust hybrid Jarratt-Butterfly optimization algorithm for nonlinear models", *Journal of King Saud University-Computer and Information Sciences,* Vol. 34, No. 10, pp. 8207-8220, 2022.