Journal of Cyber Security
DOI: 10.32604/jcs.2022.035222
Article

A Two-Tier Fuzzy Meta-Heuristic Hybrid Optimization for Dynamic Android Malware Detection

K. Santosh Jhansi[1,2,*], Sujata Chakravarty[3] and P. Ravi Kiran Varma[2]

[1]Centurion University of Technology and Management, Paralakhemundi, Odisha, India
[2]Maharaj Vijayaram Gajapathi Raj College of Engineering, Viizianagaram, India
[3]Centurion University of Technology and Management, Bhubaneswar, Odisha, India
*Corresponding Author: K. Santosh Jhansi. Email: Santosh.jhansi@gmail.com
Received: 01 September 2022; Accepted: 02 September 2022

Abstract: Application Programming Interface (API) call feature analysis is the prominent method for dynamic android malware detection. Standard benchmark android malware API dataset includes features with high dimensionality. Not all features of the data are relevant, filtering unwanted features improves efficiency. This paper proposes fuzzy and meta-heuristic optimization hybrid to eliminate insignificant features and improve the performance. In the first phase fuzzy benchmarking is used to select the top best features, and in the second phase meta-heuristic optimization algorithms viz., Moth Flame Optimization (MFO), Multi-Verse Optimization (MVO) & Whale Optimization (WO) are run with Machine Learning (ML) wrappers to select the best from the rest. Five ML methods viz., Decision Tree (DT), Random Forest (RF), K-Nearest Neighbors (KNN), Naïve Bayes (NB) & Nearest Centroid (NC) are compared as wrappers. Several experiments are conducted and among them, the best post reduction accuracy of 98.34% is recorded with 95% elimination of features. The proposed novel method outperformed among the existing works on the same dataset.

Keywords: Wrapper feature selection; multi-verse optimization; moth flame optimization; whale optimization; malware detection; classification

1 Introduction

Android smartphones and tablets are quickly gaining popularity, and with that growth has come an increase in security risks for those devices [1]. Today, risks to smart phones include the theft of user testimonials, the activation of malicious services without the user's knowledge, and service denial, among others. As a result of Android applications being susceptible to reverse engineering, attackers identify the Android-Operating System (OS) as a soft target. Malicious attackers regularly use this issue to their advantage because they frequently try to include malicious elements into legitimate apps. In contrast to other mobile OS, Android maintains openness and places little restrictions on users' ability to download and upload apps. Unfortunately, users are not the best people to evaluate the

intents of programs since they lack security awareness. Users struggle to make the best choice because they are unaware of the risks related to permissions.

As a common method for malware detection using API calls, feature selection has received considerable attention [2]. The Android app's API call vector could include up to 100 features. But the complexity of these enormous data makes handling them quite challenging. It could make learning more difficult and slow down the learning process. Because some attributes in datasets are redundant or useless, feature reduction approaches are therefore crucial for reducing the dimensionality of data. These superfluous features ought to be eliminated without lowering the categorization accuracy. The total number of all feature reduces (sets of some selected attributes) is 2^n if the number of attributes is n and the value of n is large enough. Heuristic algorithms can surely be taken into consideration because exhaustive searching methodology in this context is an NP-hard problem. As a result, the feature selection stage, in which a suitable feature subset is automatically chosen by evaluating possible feature subsets, is one of the key design processes in a pattern classifier system.

In this context, a two-tier feature selection process associating fuzzy logic [3,4] with metaheuristic optimization is explored. A hybrid model based on fuzzy optimization combined with meta-heuristic optimization algorithms viz., MVO [5], MFO [6] & WO [7–9] are examined as wrappers for its efficiency in detecting and classifying Android malware applications [10]. The main contributions on this paper are:

- Identifying problematic APIs for consistent and accurate classification of Android applications from malware and benign.
- The formulation and application of hybrid feature minimization techniques using fuzzy logic and swarm optimization.
- Using a variety of ML classifiers, such as NB, KNN, NC, RF & DT, to evaluate the suggested methodology.
- Depending on number of variables, the selection of optimal optimization algorithm for predicting Android malware.

The remainder of the paper is structured as follows: The literature review is explained in Section 2, the methodology of the proposed work is presented in Section 3, the experimental setup is described in Section 4, the performance analysis and experimentation results are explained in Section 5, and the conclusion and future work are provided in Section 6.

2 Literature Survey

Researchers worked extensively in recent years to develop methods for detecting malware on Android devices. Parnika et al. [11] demonstrated a multi-tiered feature selection approach using Information Gain. Using five different ML classifiers, they extracted the Optimal Static Feature Set (OSFS) and Most Important Features (MIFs). Their experimented methodology achieved an accuracy of 96.28% using Random Forest classifier. Qingguo et al. [12] proposed a classification model based on neural networks to distinguish between malware and benign applications. Their experimented methodology is compared with other competing machine learning classifier with various evaluation metrics and resulted in an improved accuracy of 97.85%.

Meghna et al. [13] addressed the problem of detecting benign and malware applications using imbalanced datasets. A Cost-Sensitive Forest (CSForest) consisting a set of DT with cost-sensitive voting system is explored. The evaluated technique show-cased 0.919% F-measure. Jyoti Kalita et al. [14] elucidated a framework combining MFO with Knowledge-Based-Search (KBS), considered MFO

as base optimization algorithm working beneath KBS. In conclusion, their proposed framework outperformed when compared the integrations of KBS with other optimization algorithms.

Ibrahim et al. [15] proposal of an enhanced binary MOMVO variant addressed the local optima stagnation issue of the multi-objective variant of multi-verse optimizer (MOMVO). Their experimented binary MOMVO feature selection technique outperformed compared to other methods. Julakha et al. [16] suggested a modified multi-verse optimizer, by proposing a selection mechanism based on average position to address the problem of local optima and incorporated Sine Cosine algorithm for improving the balancing mechanism of exploration and exploitation phase. The mean and standard deviations of their proposed method showed improved results when compared with different bench mark functions.

Pelusi et al. [17] introduced a fitness-based weight factor to update the position of moth between exploration and exploitation phase. Their experimentation proved that the suggested mechanism outperformed other competing optimization algorithms in terms of searching and convergence performance. By changing the update method with the help of the mutation operator and linear search, Zhao et al. [18] clarified a procedure using the Improved Moth-Flame Optimization (IMFO) algorithm. In their inquiry, orthogonal opposition-based learning controls the flame formation mechanism (OOBL). The outcomes demonstrated how the IMFO was proposed enhanced the global search algorithm.

This paper focuses on coalescing fuzzy logic with swarm optimization for effective detection & classification of android malware [19]. The most influential features to differentiate benign applications from malware are initially identified using fuzzy rule-based technique and then by wrapper-based swarm optimization feature selection methods [20,21]. The proposed feature selection process is evaluated using ML classifiers for its efficiency in detecting Android malware [22].

3 Methodology

The architecture of the proposed fuzzy based feature selection technique is highlighted in Fig. 1. The whole data of API calls is split into train and test sets at a ratio of 7:3. The features are extracted from the preprocessed dataset and fuzzy sets are generated for each feature [23]. The average of each feature is calculated and compared with a threshold value to obtain the fuzzy optimized feature set. The filtered features using fuzzification are then passed on to metaheuristic-based swarm optimized MVO, MFO & WO algorithms [24]. The derived reduced feature set is then passed on to the machine learning algorithms to evaluate the classification performance in distinguishing the malware and benign API calls of Android applications.

3.1 Fuzzy Optimized Feature Selection

A class of objects with a range of membership degrees is referred to as a fuzzy set [25]. This type of set defines a membership (characteristic) function that assigns each object a membership degree ranging between [0, 1] [26].

Let the labeled dataset be represented as $X = \{x\}$, x is an element of X. The fuzzy set A defined on X is a collection of ordered pairs, defined as shown in Eq. (1)

$$A = \{x, \mu_A(x; a,\ b,\ c,\ d)\}, x \in X \tag{1}$$

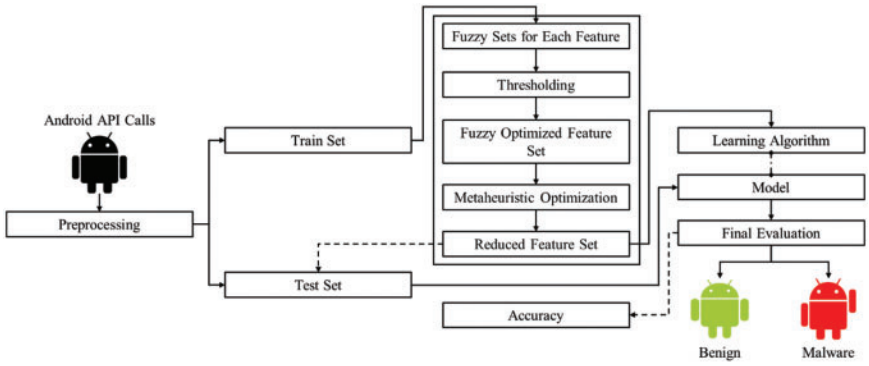

Figure 1: Architecture of proposed android malware classification system

The projection of each element x into fuzzy space is carried out using a trapezoidal membership function: $\mu_A(x; a,\ b,\ c,\ d)$, which is defined as shown in Eqs. (2) and (3)

$$\mu_{trapezoidal}(x; a,\ b,\ c,\ d) = \begin{cases} 0, & x \le a \\ \dfrac{x-a}{b-a}, & a \le x \le b \\ 1, & b \le x \le c \\ \dfrac{d-x}{d-c}, & c \le x \le d \\ 0, & d \le x \end{cases} \tag{2}$$

$$= max\left(min\left(\frac{x-a}{b-a},\ 1\ , \frac{d-x}{d-c}\right), 0\right) \tag{3}$$

Here, $a,\ b,\ c$ & d represents the quantile values of feature as (0.100, 0.25, 0.50 & 0.75) respectively. For each feature of the obtained fuzzy set, standard deviation is calculated and is represented as shown in Eq. (4)

$$Z = [z_1,\ z_2,\ z_3,\ \ldots,\ z_n] \tag{4}$$

where, n is the total number of features. Each feature of Z satisfying the below condition shown in Eq. (5) is considered as optimized feature. Here, $T \in [0,\ 1]$ is a threshold value.

$$Z_i > T \tag{5}$$

3.2 Wrapper-Based Multi-Verse Optimized Feature Selection (WMVOFS)

Mirjalili introduced the Multi-Verse Optimizer in 2015. The algorithm used in the MVO method was motivated by physics [27]. Worm hole, black hole, and white hole are the three cosmological principles on which this method is defined. Multiverse is the antithesis of universe, which implies the existence of universes other than the one we all currently inhabit.

Optimization Rules of Multi-Verse Optimizer:

- The likelihood of a white hole is higher when the inflation rate is higher.
- More matter passes through black holes in universes with lower inflation rates.
- Every universe contains elements that can cause random movement in the direction of the universe that is most suitable. A black hole is less likely to exist if the inflation rate is higher.

- White holes are used to transmit material through universes with higher inflation rates.
- Worm holes occur when objects travel from a universe with a higher inflation rate to one with a lower inflation rate, regardless of the inflation rate.

In each iteration, the universes are sorted according to its inflation rates and select one from them using the roulette wheel as a white hole. The following steps are done in order to achieve this:

Assume,

$$U = \begin{bmatrix} x_1^1 & x_1^2 & \dots & x_1^d \\ x_2^1 & x_2^2 & \dots & x_2^d \\ . & . & \dots & . \\ . & . & \dots & . \\ . & . & \dots & . \\ x_n^1 & x_n^2 & \dots & x_n^d \end{bmatrix} \tag{6}$$

$$x_i^j = \begin{cases} x_k^j \; r1 \le NI\,(Ui) \\ x_i^j \; r1 \ge NI\,(Ui) \end{cases} \tag{7}$$

where, d defines the number of parameters and n defines the number of universes, x_i^j is the j^{th} parameter of i^{th} universe, Ui is the i^{th} universe, $NI\,(Ui)$ is the normalized inflation rate of i^{th} universe, $r1$ is a random number in [0, 1], x_k^j indicates the j^{th} parameter of k^{th} universe selected by a roulette wheel selection mechanism.

Algorithm 1: WMVOFS

Random Universe (U) creation
$WEP,\ TDR$ & $Best\_universe$ initialization
SU = Sorted universes
NI = Normalize the inflation rate (*fitness*) of the universe
While *end criterion* is not satisfied
    Fitness evaluation of all universes
    For each *universe i*
        Updation of WEP & TDR
        $Black_{hole_{index}} = i$
        For each *object j*
            $r1$ = random number
            If $r1 < NI\,(Ui)$
                $White_{hole_{index}} = RouletteWheelSeleciton\,(-NI)$
                $U\left(Black_{hole_{index}},\ j\right) = SU\left(White_{hole_{index}},\ j\right)$
            End if
            $r2$ = random number
            If $r2 < WEP$
                $r3,\ r4$ = random number, random number
            If $r3 < 0.5$
                $U\,(i,j) = Best\_universe\ +\ \text{TDR} * ((ub_j - lb_j) * \left(r4 + lb_j\right))$

(Continued)

Algorithm 1: Continued

```
                                    Else
                                    U (i, j) = Best_universe − TDR * ((ub_j − lb_j) ∗ (r4 + lb_j))
                            End if
                            End if
                    End for
            End for
End while
```

The likelihood of sending objects through white or black hole tunnels increases with decreasing inflation rate. The universe is currently swapping objects without interruption. Each universe is thought to include worm holes that randomly transfer its things around space in order to preserve the diversity of the universe and carry out exploitation. Assume that wormhole tunnels are always created between a universe and the best universe that has yet to be created, as shown by Eq. (8).

$$x_i^j = \begin{cases} \begin{cases} X_j + \text{TDR} * ((ub_j - lb_j) * (r4 + lb_j) & r3 < 0.5,\ r2 < WEP \\ X_j - \text{TDR} * ((ub_j - lb_j) * (r4 + lb_j) & r3 \geq 0.5,\ r2 \geq WEP \end{cases} \\ x_j^i \end{cases} \tag{8}$$

where, X_j is the j^{th} parameter of best universe formed so far, lb_j is the lower bound of j^{th} variable, ub_j is the upper bound of j^{th} variable, $r2,\ r3$ & $r4$ are random number in [0, 1]. Worm hole Existence Probability (WEP) is required to increase linearity over the iteration in order to emphasize exploitation as the progress of optimization process. It is calculated using Eq. (9)

$$WEP = \min + I \times \left(\frac{max - min}{L}\right) \tag{9}$$

where, I is the current iteration and L is the maximum number of iterations. Using Eq. (10) it is possible to determine the distance rate (variation) at which an object can travel through the best universe.

$$Travelling\ Distance\ Rate\ \ (TDR) = 1 - \frac{l^{1/p}}{L^{1/p}} \tag{10}$$

The larger the, p value, the quicker and more accurate the exploitation/local search over the iterations.

3.3 Wrapper-Based Moth-Flame Optimized Feature Selection (WMFOFS)

The Moth-Flame optimizer was also introduced by Mirjalili [28]. This optimizer was primarily inspired by moths' natural transverse orientation navigational strategy. Moths use an efficient method of long-distance straight-line movement at night by maintaining a stable angle in relation to the moon. These elegant insects are, however, entangled in a pointless or fatal spiral journey around artificial lights. To conduct optimization, the MFO method simulates this behavior mathematically.

Here, l defines current iteration number, N indicates the maximum number of flames, T introduces the maximum number of iterations, D_i represents the i^{th} moth for j^{th} flame, b indicates the shape of logarithmic spiral constant, t defines random number $\epsilon\ [-1, 1]$, M_i defines the i^{th} moth, F_j represents the j^{th} flame and r is a variable linearly decreasing from -1 to -2 during the course of iteration.

Algorithm 2: WMFOFS

Update the no. of flames using $flameno = round\left(N - l * \frac{N-1}{T}\right)$
Moth's population initialization
Defining objective function
For all *moths i*
    For all *parameters j*
        Update t and r
        Calculate D using $D_i = \left|F_j - M_i\right|$ w.r.t corresponding moth
    Update matrix M using $M_i = S(M_i, F_i)$ & $S(M_i, F_i) = D_i.e^{bt}.Cos(2\pi t) + F_j$ w.r.t corresponding moth
    End for
    Calculate objective values
    Update flames
End for

3.4 Wrapper-Based Whale Optimized Feature Selection (WWOFS)

WO algorithm introduced by Seyedali Mirjalili, mimics the bubble-net attack strategy used by humpback whales when attacking prey [29]. The ideal solution is regarded as the best available candidate. regarding the best candidate solution, the remaining whales adjust their positions. The three stages of this techniques are the encircle the prey phase, the exploration phase, and the search phase.

Where, $\vec{X}$ represents the whale's position vector, t defines iterations, $\vec{X}_{best}$ explains best whale's position vector, $\vec{A}$ & $\vec{C}$ are co-efficient vectors calculated using Eqs. (11) and (12), b interprets a constant and $\vec{X}_{random}$ is a random number in [0, 1].

$$\vec{A} = 2 * \vec{a} * \vec{r1} - \vec{a} \tag{11}$$

$$\vec{C} = 2 * \vec{r2} \tag{12}$$

where, $\vec{r1}$, $\vec{r2} \in [0, 1]$ are random vectors and $\vec{a}$ is a linear component decreasing from 2 to 0.

Algorithm 3: WWOFS

Initialization of whale population
Fitness of whale is calculated
Determination of best whale
While end criterion not obtained
    For each whale
        Update: a, A, C, l & p
        If $p < 0.5$
            If $|A| < 1$
                $\vec{D} = \left|\vec{C} \cdot \vec{X}_{best}(t) - \vec{X}(t)\right|$
            Else
                $\vec{X}(t+1) = \vec{X}_{random}(t) - \vec{A}.\vec{D}$
            End if
        Else

(Continued)

Algorithm 3: Continued

$$\vec{D} = \left|\vec{C} \cdot \vec{X}_{random}(t) - \vec{X}(t)\right|$$

$$\vec{X}(t+1) = \vec{D}^{l}.e^{bl}.\cos(2\pi l) + \vec{X}_{best}(t)$$

End if

End for

Update best whale

End while

4 Experimental Setup

The overall experimentation was executed on a Intel® Core™ i^5 processor clocked at 1.80 GHz, Windows 10 (64-bit) operating system with an, 4 GB of RAM, and a 2 TB hard drive installed with Anaconda, a Python platform with machine learning supporting packages as its setup.

Data on API call sequences for the experiment was gathered through the IEEE Data Port. They are 100 features and 43,876 samples out of which 42,797 are malware, while 1,079 of them are benign applications [30]. Data for the experiment is gathered using the Cuckoo Sandbox environment, and it is then confirmed with Virus Total.

5 Performance Analysis and Experimental Results

Through a variety of classification analysis measures, including Root Mean Square Error (RMSE), Mean Squared Error (MSE), Precision, Recall, F1-Score, and Accuracy, the suggested approach for Android malware detection verifies the classification accuracy.

$$Precision = \frac{True_{pos}}{False_{pos} + True_{pos}} \tag{13}$$

$$Recall = \frac{True_{pos}}{False_{neg} + True_{pos}} \tag{14}$$

$$F1 - Score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{15}$$

$$Accuracy = \frac{True_{pos} + True_{neg}}{True_{pos} + False_{pos} + True_{neg} + False_{neg}} \tag{16}$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2 \tag{17}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2} \tag{18}$$

where, $True_{pos}$ characterizes the samples identified correctly as good ware, $True_{neg}$ represents the samples identified correctly as malware, $False_{pos}$ is the samples incorrectly categorized as goodware, $False_{neg}$ is the samples incorrectly described as malware, $\hat{Y}_i$ is the predicted output, Y_i is the actual output, and n defines the number of samples. On the API calls sequence dataset, the MVO, MFO, and WO algorithms are examined for their performance when wrapped with KNN, NC, DT, NB, and RF classifiers using the suggested methodology. The free parameter used for all the experiments are described in Table 1. The results of the experimentation conducted using different thresholds (0.41,

0.42, 0.43, 0.44 & 0.45) for fuzzy optimized feature selection followed by wrapped-based feature selection are listed from Tables 2–6. Their graphical illustrations are shown from Figs. 2–6.

Table 1: Free parameters of MVO, MFO & WO

MVO	MFO	WO
Iterations = 20	Iterations = 20	Iterations = 20
Lb = −1	Lb = −1	Lb = −1
Ub = 1	Ub = 1	Ub = 1
WEP_MAX = 1	N = 20	Search_Agents_No = 20
WEP_MIN = 0.2		
N = 20		

Table 2: Accuracy comparison of MVO, MFO & WAO wrapped with KNN

S. No	Accuracy before feature selection	Threshold	Fuzzy optimized features count	Optimizer	Features selected	Accuracy after feature selection	% change in accuracy	% decrease in features
1	98.40%	0.41	62	MVO	18	98.36%	−0.0400%	82%
				MFO	26	98.40%	0.0000%	74%
				WOA	21	98.40%	0.0000%	79%
2		0.42	44	MVO	15	98.40%	0.0000%	85%
				MFO	16	98.25%	−0.1500%	84%
				WOA	13	98.42%	0.0200%	87%
3		0.43	28	MVO	8	98.22%	−0.1800%	92%
				MFO	12	98.40%	0.0000%	88%
				WOA	**5**	**98.34%**	**−0.0600%**	**95%**
4		0.44	15	MVO	4	98.27%	−0.1300%	96%
				MFO	7	98.27%	−0.1300%	93%
				WOA	4	98.25%	−0.1500%	96%
5		0.45	8	MVO	1	97.32%	−1.0800%	99%
				MFO	1	97.47%	−0.9300%	99%
				WOA	1	97.32%	−1.0800%	99%

Table 3: Accuracy comparison of MVO, MFO & WAO wrapped with NC

S. No	Accuracy before feature selection	Threshold	Fuzzy optimized features count	Optimizer	Features selected	Accuracy after feature selection	% change in accuracy	% decrease in features
1	78.94%	0.41	62	MVO	42	81.78%	2.8400%	58%
				MFO	44	83.36%	4.4200%	56%
				WOA	30	82.47%	3.5300%	70%
2		0.42	44	MVO	25	79.17%	0.2300%	75%
				MFO	26	78.54%	−0.4000%	74%
				WOA	35	78.53%	−0.4100%	65%
3		0.43	28	MVO	18	80.37%	1.4300%	82%
				MFO	17	78.27%	−0.6700%	83%
				WOA	20	78.57%	−0.3700%	80%
4		0.44	15	**MVO**	**6**	**76.44%**	**−2.5000%**	**94%**
				MFO	11	74.99%	−3.9500%	89%
				WOA	7	74.11%	−4.8300%	93%
5		0.45	8	MVO	4	76.34%	−2.6000%	96%
				MFO	4	76.34%	−2.6000%	96%
				WOA	4	76.34%	−2.6000%	96%

Table 4: Accuracy comparison of MVO, MFO & WAO wrapped with DT

S. No	Accuracy before feature selection	Threshold	Fuzzy optimized features count	Optimizer	Features selected	Accuracy after feature selection	% change in accuracy	% decrease in features
1	98.42%	0.41	62	MVO	25	98.18%	−0.2400%	75%
				MFO	29	98.11%	−0.3100%	71%
				WOA	18	98.12%	−0.3000%	82%
2		0.42	44	MVO	15	98.08%	−0.3400%	85%
				MFO	18	98.03%	−0.3900%	82%
				WOA	10	98.12%	−0.3000%	90%
3		0.43	28	MVO	8	98.12%	−0.3000%	92%
				MFO	7	98.12%	−0.3000%	93%
				WOA	8	98.12%	−0.3000%	92%
4		0.44	15	MVO	**4**	**98.02%**	**−0.4000%**	**96%**
				MFO	6	98.02%	−0.4000%	94%
				WOA	4	97.71%	−0.7100%	96%
5		0.45	8	MVO	1	97.44%	−0.9800%	99%

(Continued)

Table 4: Continued

S. No	Accuracy before feature selection	Threshold	Fuzzy optimized features count	Optimizer	Features selected	Accuracy after feature selection	% change in accuracy	% decrease in features
				MFO	1	97.44%	−0.9800%	99%
				WOA	1	97.44%	−0.9800%	99%

Table 5: Accuracy comparison of MVO, MFO & WAO wrapped with NB

S. No	Accuracy before feature selection	Threshold	Fuzzy optimized features count	Optimizer	Features selected	Accuracy after feature selection	% change in accuracy	% decrease in features
1	89.06%	0.41	62	MVO	27	98.18%	9.1200%	73%
				MFO	27	98.09%	9.0300%	73%
				WOA	19	97.36%	8.3000%	81%
2		0.42	44	MVO	15	98.14%	9.0800%	85%
				MFO	17	97.44%	8.3800%	83%
				WOA	15	98.19%	9.1300%	85%
3		0.43	28	MVO	10	97.45%	8.3900%	90%
				MFO	10	97.67%	8.6100%	90%
				WOA	9	98.19%	9.1300%	91%
4		0.44	15	MVO	7	97.42%	8.3600%	93%
				MFO	9	97.43%	8.3700%	91%
				WOA	**8**	**97.43%**	**8.3700%**	**92%**
5		0.45	8	MVO	1	97.43%	8.3700%	99%
				MFO	1	97.43%	8.3700%	99%
				WOA	1	97.43%	8.3700%	99%

Table 6: Accuracy comparison of MVO, MFO & WO wrapped with RF

S. No	Accuracy before feature selection	Threshold	Fuzzy optimized features count	Optimizer	Features selected	Accuracy after feature selection	% change in accuracy	% decrease in features
1	98.92%	0.41	62	MVO	25	98.61%	−0.3100%	75%
				MFO	26	98.62%	−0.3000%	74%
				WOA	25	98.64%	−0.2800%	75%

(Continued)

Table 6: Continued

S. No	Accuracy before feature selection	Threshold	Fuzzy optimized features count	Optimizer	Features selected	Accuracy after feature selection	% change in accuracy	% decrease in features
2		0.42	44	MVO	13	98.61%	−0.3100%	87%
				MFO	14	98.62%	−0.3000%	86%
				WOA	13	98.64%	−0.2800%	87%
3		0.43	28	MVO	10	98.61%	−0.3100%	90%
				MFO	7	98.50%	−0.4200%	93%
				WOA	8	98.60%	−0.3200%	92%
4		0.44	15	MVO	5	98.49%	−0.4300%	95%
				MFO	**6**	**98.58%**	**−0.3400%**	**94%**
				WOA	5	98.55%	−0.3700%	95%
5		0.45	8	MVO	3	97.73%	−1.1900%	97%
				MFO	3	97.73%	−1.1900%	97%
				WOA	1	97.53%	−1.3900%	99%

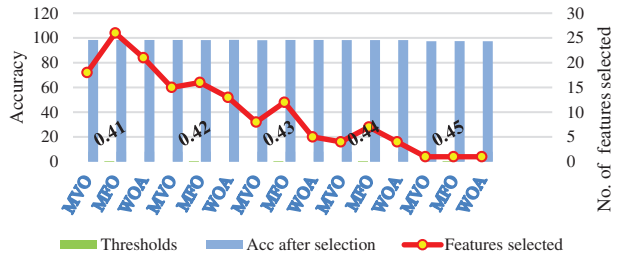

Figure 2: Performance comparison of MVO, MFO & WOA wrapped with KNN

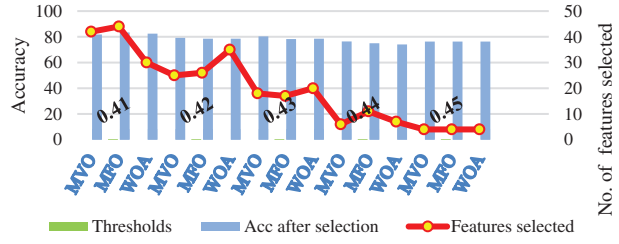

Figure 3: Performance comparison of MVO, MFO & WOA wrapped with NC

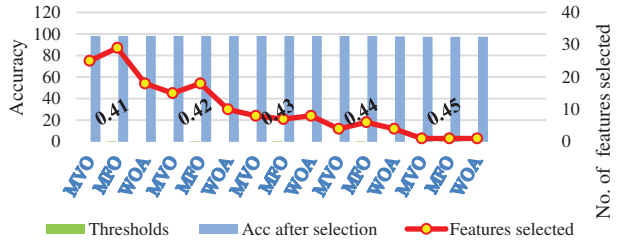

Figure 4: Performance comparison of MVO, MFO & WOA wrapped with DT

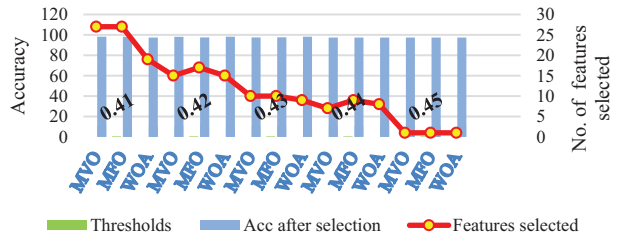

Figure 5: Performance comparison of MVO, MFO & WOA wrapped with NB

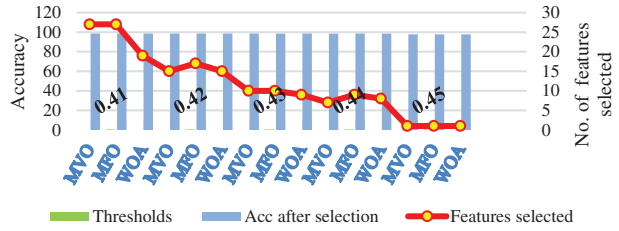

Figure 6: Performance comparison of MVO, MFO & WOA wrapped with RF

The experimental results indicate that the KNN classifier when wrapped with WOA resulted in an accuracy of 98.34% with 95% reduction of feature set. The NC classifier when wrapped with MVO obtained an accuracy of 76.44% having 94% reduced feature set. The DT classifier when wrapped with MVO achieved an accuracy of 98.02% while maintaining 96% reduction in feature set dimensionality. NB classifier when wrapped with WOA obtained an accuracy of 97.43% with 92% reduction in dimensionality of feature set. Similarly, the RF classifier when wrapped with MFO resulted in an accuracy of 98.58% with 94% reduction in dimensionality of feature set.

The 5 features selected by Wrapper-Based Whale Optimized Feature Selection using KNN classifier are listed in Table 7. For the WWOFS algorithm wrapped with KNN, DT, NB, NC & RF classifiers, the Area Under Curve Receiver Operator Characteristic (AUC_ROC) graphs are generated, as this algorithm outperformed the comparative algorithms. The WWOFS embedded with KNN has a lesser area under the AUC_ROC curve when used with a smaller feature set compared to the area under the full feature set. The AUC_ROC graphs for each of the tested ML classifiers are shown in Figs. 7–11. The accuracy comparison of related work the proposed method is presented in Table 8.

Table 7: APIs selected by WMVOFS using DT

S. No	API No	API description
1	5	LdrGetProcedureAddress
2	12	LdrLoadDll
3	27	RegQueryValueExW
4	89	LdrGetDllHandle
5	93	NtClose

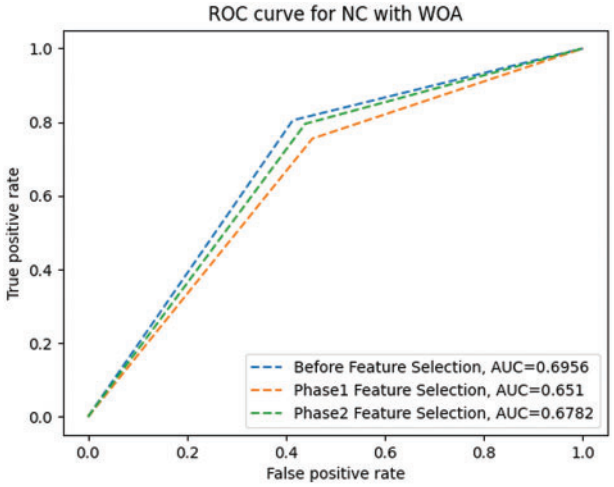

Figure 7: AUC_ROC curve of WWOFS + NC

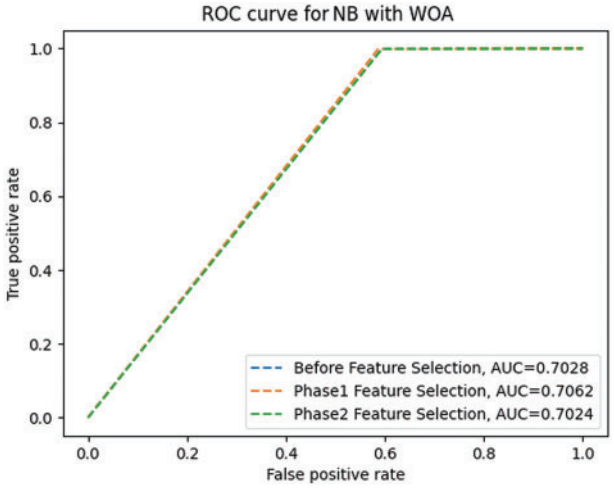

Figure 8: AUC_ROC curve of WWOFS + NB

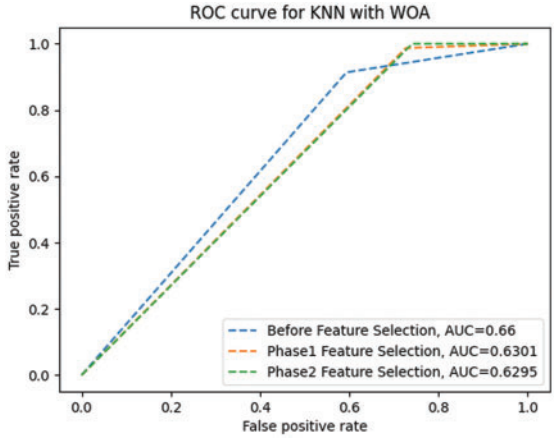

Figure 9: AUC_ROC curve of WWOFS + KNN

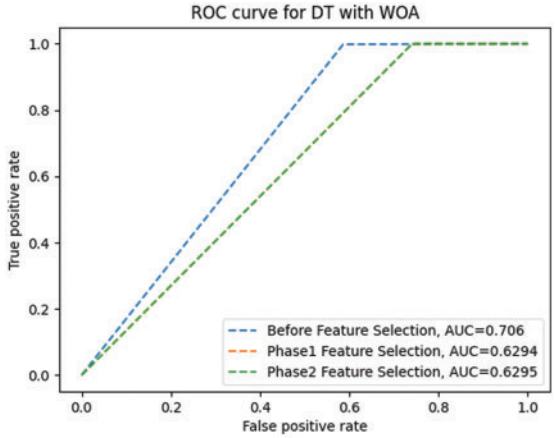

Figure 10: AUC_ROC curve of WWOFS + DT

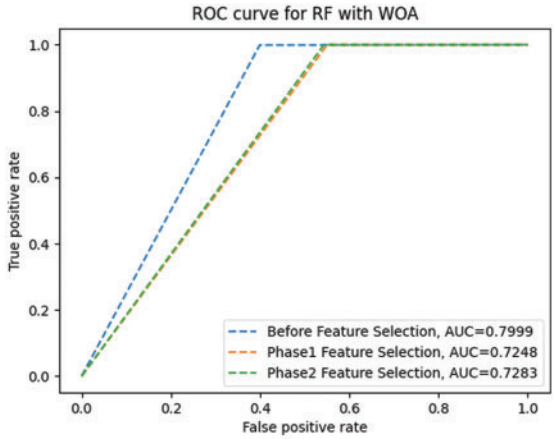

Figure 11: AUC_ROC curve of WWOFS + RF

Table 8: Accuracy comparison of related work

Paper, Year	Classifier	Feature selection	Accuracy
[30], 2019	Deep Graph Convolutional Neural Networks (DGCNN)	-	92.44%
[31], 2022	Neural Oblivious Decision Ensembles (NODE)	-	90%
This paper	KNN	WWOFS	98.53%

6 Conclusion and Future Work

This paper examines a hybrid feature minimization approach for attribute space using a fuzzy logic and swarm optimization. Initially, complete feature set is optimized using fuzzy logic. To identify the most influential feature space, the obtained features are fed into wrapper-based feature selection process. The narrowed feature set is then used to train the machine learning model to differentiate between benign and malware Android applications. The wrapper-based feature selection techniques such as WMVOFS, WMFOFS & WWOFS reduced the dimensionality of the feature space to a greater extent when incorporated with fuzzy optimized feature set.

Out of MVO, MFO & WO, the WO when wrapped with ML algorithms outperformed other competing algorithms in terms of minimizing the dimensionality of feature space. The WWOFS algorithm when wrapped with KNN classifier achieved better results in reducing the feature space to 95% while maintaining an accuracy of 98.53%.

Future work will involve the development and application of hybrid architectures with advanced deep learning techniques for better efficiency in the detection and classification of Android malware. It will also involve investigating other optimization algorithms for systematized feature reduction using high-dimensional feature space.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] M. Talal, A. A. Zaidan, B. B. Zaidan and O. S. Albahri, "Comprehensive review and analysis of anti-malware apps for smartphones," *Telecommunication Systems*, vol. 1, no. 72, pp. 285–337, 2019.

[2] A. Bhattacharya, R. T. Goswam and K. Mukherjee, "A feature selection technique based on rough set and improvised PSO algorithm (PSORS-FS) for permission-based detection of android malwares," *International Journal of Machine Learning and Cybernetics*, vol. 10, pp. 1893–1907, 2019.

[3] V. Lakovic, "Crisis management of android botnet detection using adaptive neuro-fuzzy inference system," *Annals of Data Science*, vol. 7, pp. 1–4, 2020.

[4] B. Saridou, J. R. Rose, S. Shiaeles and P. Basil, "SAGMAD—A signature agnostic malware detection system based on binary visualization and fuzzy sets," *Electronics*, vol. 11, no. 7, pp. 1–26, 2022.

[5] P. C. Sahu, B. S. Kumar, J. N. Kumar and B. K. Sahu, "A robust multi verse optimized fuzzy aided tilt controller for AGC of hybrid power system," in *Odisha Int. Conf. on Electrical Power Engineering, Communication and Computing Technology (ODICON)*, Bhubaneswar, India, 2021.
[6] G. Deepak, K. A. Anil, A. Sharma and J. J. P. C. Rodrigues, "Feature selection and evaluation for software usability model using modified moth-flame optimization," *Computing*, vol. 102, pp. 1503–1520, 2020.
[7] Y. Zheng, Y. Li, G. Wang, Y. Chen, Q. Xu *et al.,* "A novel hybrid algorithm for feature selection based on whale optimization algorithm," *IEEE Access*, vol. 7, pp. 14908–14923, 2019.
[8] P. Ravi Kiran Varma, M. S. K. Reddy, K. S. Jhansi and D. Pushpa Latha, "Bat optimization algorithm for wrapper-based feature selection and performance improvement of android malware detection," *IET Networks*, vol. 10, pp. 131–140, 2021.
[9] X. Chen, "Research on new adaptive whale algorithm," *IEEE Access*, vol. 8, pp. 90165–90201, 2020.
[10] K. Santosh Jhansi, P. Ravi Kiran Varma and S. Chakravarty, "Swarm optimization and machine learning for android malware detection," *Computers, Materials & Continua* (In Press), 2022.
[11] B. Parnika and D. Kamlesh, "A multi-tiered feature selection model for android malware detection based on feature discrimination and information gain," *Journal of King Saud-University-Computer and Information Sciences* (In Press), 2021.
[12] Z. Qingguo, F. Fang, S. Zebang, Z. Rui, M. Y. Hsieh *et al.,* "A novel approach for mobile malware classification and detection in android systems," *Multimedia Tools and Applications*, vol. 78, pp. 3529–3552, 2019.
[13] D. Meghna and G. Ekta, "CSForest: An approach for imbalanced family classification of android malicious applications," *International Journal of Information Technology*, vol. 13, pp. 1059–1071, 2021.
[14] D. Jyoti Kalita, V. Prakash Singh and K. Vinay, "A dynamic framework for tuning SVM hyper parameters based on moth-flame optimization and knowledge-based-search," *Expert Systems with Applications*, vol. 168, pp. 1–47, 2021.
[15] A. Ibrahim, H. Faris, A. H. Asghar, M. M. Mafarja, A. M. Al-Zoubi *et al.,* "A robust multi-objective feature selection model based on local neighborhood multi-verse optimization," *IEEE Access*, vol. 9, pp. 100009–100028, 2021.
[16] J. J. Julakha, M. S. Ahmad and M. I. M. Rashid, "Modified multi-verse optimizer for solving numerical optimization problems," in *2020 IEEE Int. Conf. on Automatic Control and Intelligent Systems (I2CACIS)*, Shah Alam, Malaysia, 2020.
[17] D. Pelusi, M. Raffale, T. Luca, J. Nayak, B. Naik *et al.,* "An improved moth-flame optimization algorithm with hybrid search phase," *Knowledge-Based Systems*, vol. 191, pp. 1–43, 2020.
[18] X. Zhao, Y. Fang, L. Liu, J. Li and X. Miao, "An improved moth-flame optimization algorithm with orthogonal opposition-based learning and modified position updating mechanism of moths for global optimization problems," *Applied Intelligence*, vol. 50, pp. 4434–4458, 2020.
[19] M. Arvind and A. L. Sangal, "HybriDroid: An empirical analysis on effective malware detection model developed using ensemble methods," *The Journal of Supercomputing*, vol. 77, no. 1, pp. 1–44, 2021.
[20] T. Franklin, C. N. N. Roger, V. C. Kamla and K. U. Priyanath, "LimonDroid: A system coupling three signature-based schemes for profiling android malware," *Iran Journal of Computer Science*, vol. 4, pp. 95–114, 2021.
[21] A. Prerna and H. T. Bhushan, "Machine learning classifiers for android malware detection," in *Data Management, Analytics and Innovation*, vol. 1174. Singapore: Springer, pp. 311–322, 2021.
[22] J. Xiao, Q. Han and Y. Gao, "Hybrid classification and clustering algorithm on recent android malware detection," in *2021 5th Int. Conf. on Computer Science and Artificial Intelligence (CSAI 2021)*, Beijing China, 2021.
[23] A. Taha and S. Malebary, "Hybrid classification of android malware based on fuzzy clustering and the gradient boosting machine," *Neural Computing and Applications*, vol. 33, pp. 6721–6732, 2020.
[24] S. Sapre and S. Mini, "Emulous mechanism based multi-objective moth–flame optimization algorithm," *Journal of Parallel and Distributed Computing*, vol. 150, pp. 15–33, 2021.

[25] P. Sanki, M. Surjakanta, M. Basu, P. S. Pal and D. Das, "Moth flame optimization based fuzzy-PID controller for power–frequency balance of an islanded microgrid," *Journal of the Institution of Engineers (India): Series B*, vol. 102, pp. 997–1006, 2021.
[26] X. Liu, X. Du, Q. Lei and K. Liu, "Multifamily classification of android malware with a fuzzy strategy to resist polymorphic familial variants," *IEEE Access*, vol. 8, pp. 156900–156914, 2020.
[27] S. Mirjalili, M. Mohmmad and A. Hatamlou, "Multi-verse optimizer: A nature-inspired algorithm for global optimization," *Neural Computing and Applications Volume*, vol. 27, pp. 495–513, 2016.
[28] S. Mirjalili, "Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm," *Knowledge-Based Systems*, vol. 89, pp. 228–249, 2015.
[29] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Advances in Engineering Software*, vol. 95, pp. 51–67, 2016.
[30] A. Oliveira and S. R. Jose, "Behavioral malware detection using deep graph convolutional neural networks," *International Journal of Computer Applications*, vol. 174, no. 29, pp. 1–8, 2021.
[31] A. Cannarile, V. Dentamaro, S. S. Galantucci, A. A. Iannacone, D. lmpedovo *et al.,* "Comparing deep learning and shallow learning techniques for API calls malware prediction: A study," *Applied Sciences*, vol. 12, no. 3, pp. 1–16, 2022.