Enhanced Android Ransomware Detection Through Hybrid Simultaneous Swarm-Based Optimization

Moutaz Alazab[1] · Ruba Abu Khurma[2,3] · David Camacho[4] · Alejandro Martín[4]

Abstract

Ransomware is a significant security threat that poses a serious risk to the security of smartphones, and its impact on portable devices has been extensively discussed in a number of research papers. In recent times, this threat has witnessed a significant increase, causing substantial losses for both individuals and organizations. The emergence and widespread occurrence of diverse forms of ransomware present a significant impediment to the pursuit of reliable security measures that can effectively combat them. This constitutes a formidable challenge due to the dynamic nature of ransomware, which renders traditional security protocols inadequate, as they might have a high false alarm rate and exert significant processing demands on mobile devices that are restricted by limited battery life, CPU, and memory. This paper proposes a novel intelligent method for detecting ransomware that is based on a hybrid multi-solution binary JAYA algorithm with a single-solution simulated annealing (SA). The primary objective is to leverage the exploitation power of SA in supporting the exploration power of the binary JAYA algorithm. This approach results in a better balance between global and local search milestones. The empirical results of our research demonstrate the superiority of the proposed SMO-BJAYA-SA-SVM method over other algorithms based on the evaluation measures used. The proposed method achieved an accuracy rate of 98.7%, a precision of 98.6%, a recall of 98.7%, and an F1 score of 98.6%. Therefore, we believe that our approach is an effective method for detecting ransomware on portable devices. It has the potential to provide a more reliable and efficient solution to this growing security threat.

Keywords Malware · Ransomware · Optimzation · Simulated annealing · SMOTE · Feature selection

Introduction

The Android operating system is a popular target for hackers due to a variety of factors. One key reason is the sheer size of the market, and with billions of Android devices in use around the world [1], there is a vast potential audience for malicious attacks. The open-source nature of the Android means that there is a wide range of potential vulnerabilities that hackers can exploit. Finally, the diversity of hardware and software configurations among Android devices makes it difficult to create security patches that are universally applicable, leaving many devices vulnerable to malware and other types of attacks. Malware encompasses a variety of forms such as viruses, worms, Trojans, and ransomware, which can be used to compromise mobile devices and networks, steal sensitive data, and extort victims through encryption and ransom demands.

Ransomware attacks date back to the late 1980s [2], but they have become increasingly sophisticated and widespread

✉ Moutaz Alazab
m.alazab@bau.edu.jo

Ruba Abu Khurma
ruba_abukhurma@yahoo.com

David Camacho
david.camacho@upm.es

Alejandro Martín
alejandro.martin@upm.es

1 Faculty of Artificial Intelligence, Department of Intelligent Systems, Al-Balqa Applied University, Al-Salt, Jordan

2 MEU Research Unit, Faculty of Information Technology, Middle East University, Amman 11831, Jordan

3 Applied Science Research Center, Applied Science Private University, Amman 11931, Jordan

4 Department of Computer Systems Engineering, Universidad Politécnica de Madrid, Madrid, Spain

in recent years. The first known ransomware attack was known as the AIDS Trojan, which targeted victims by displaying a message claiming that their computer was locked due to illegal activity and could only be unlocked by sending payment to an address in Panama [2]. According to the 2022 Cyber Threat Report,[1] ransomware attack has risen to 105% worldwide. Ransomware spreads malicious code through smart cell devices to prevent its users from accessing data or even interacting with the system until a payment is made [3]. Ransomware uses many methods for financial extortion, causing massive financial damage to individuals or even businesses. The emergence of novel ransomware variants such as BadRabbit, BitPayer, Cerber, Cryptolocker, Dharma, DoppelPaymer, GandCrab, and Locky has resulted in substantial economic damages, with losses reaching millions of dollars [4].

The exponential rise of ransomware attacks on smartphones presents a critical challenge to digital security. The widespread adoption of smartphones has amplified the potential consequences of ransomware, inflicting significant financial losses on individuals and businesses in both personal and professional spheres. While research efforts dedicated to combating this problem are ongoing, the current landscape paints a concerning picture. Traditional anti-ransomware security protocols often produce high rates of false alarms and require high computational speed. This dynamic threat environment reaffirms the critical importance of novel security solutions which are able to predict and detect ransomware attacks on mobile phones effectively. Since ransomware keeps changing rapidly and mobile platforms are limited in resources, the currently available solutions lack completeness in protection. Thus, the creation of new detection methods is essential to increase our resistance against those kinds of attacks.

Several security preventive controls have been enforced on smartphones to control or prevent the distribution of ransomware on Android platforms. The countermeasures comprise malware detection, vulnerability detection, and application strengthening. Malware detection techniques are the most widespread among these and can be categorized into three groups: static, dynamic, and hybrid methods as discussed in [5]. Static methods extract features such as permissions and API calls from the Android application package (APK) without executing the source code, while dynamic methods implement the APK on an isolated environment to examine the malware's behavior. Hybrid methods combine static and dynamic methods to increase their effectiveness by concurrently analyzing the application's static features and code execution. In general, two primary methods are utilized for malware detection: signature-based and anomaly-based. Signature-based methods compare the attributes of a threat with those of previously known threats in a database. Although these methods are effective in detecting known threats, they are unable to detect newly emerging threats. Anomaly-based methods do not depend on the presence of a database and are, therefore, more effective in detecting new malware species by predicting any deviation from normal network behavior. These techniques are essential for preventing ransomware and other malicious activities on smartphones, thereby improving the security of these devices.

Machine learning techniques have gained popularity in recent times and replaced traditional methods as they have demonstrated outstanding performance in malware detection [6]. The essential aspect in detecting malware is to select pertinent and non-redundant features from a pool of malware features. Meta-heuristic algorithms are often employed to randomly search and acquire the best solution within a reasonable running time [5]. The metaheuristic algorithm search process comprises exploration and exploitation. Exploration entails searching throughout the search area until the region with the best solution potential is determined. Exploitation involves searching in a promising area and enhancing the best solution's quality until a location closer to the optimal solution is attained. Meta-heuristic algorithms can be population-based (multi-solutions) which are exploration-based methods or trajectory-based (single-solution) which are exploitation-based. JAYA algorithm [7] is a population-based algorithm that uses general common parameters, thereby eliminating the need for algorithm-specific parameters (parameter-less algorithm). This paper utilizes the binary JAYA algorithm to search globally for the optimal Android malware features subset. The proposed method aims to detect ransomware using the support vector machine (SVM) algorithm. The JAYA algorithm explores the feature space to identify and optimize the number of features. Simulated annealing (SA) is a trajectory algorithm employed to enhance the final solution's quality [8].

The limited availability of ransomware datasets can be attributed to the difficulties associated with acquiring and assessing ransomware samples. These challenges arise from the intricate encryption and obfuscation methods employed by ransomware, which impede effective detection and classification. Consequently, an imbalanced class distribution emerges, with the non-ransomware category being predominant and the ransomware class being rare [9]. This results in additional hurdles for the support vector machine (SVM) during the learning phase, as the classifier will tend to favor the dominant class, leading to the overfitting issue. To address this issue, the synthetic minority oversampling technique (SMOTE) is employed to balance the two classes. The JAYA algorithm is employed to identify the optimal feature subset, the optimal number of nearest neighbors, and the ideal SMOTE sampling ratio. The aim of this study is to develop

[1] https://fortune.com/2022/02/17/ransomware-attacks-surge-2021-report/.

a high-performance ransomware detection system using a highly imbalanced dataset. The collected imbalanced dataset represents the current market, where the percentage of ransomware is low in comparison to regular applications [10].

Motivation and Significance

This paper proposes a novel and groundbreaking approach for detecting ransomware on smartphones and other portable devices. Our method, referred to as SMO-BJAYA-SA-SVM, leverages the combined strengths of two powerful algorithms: JAYA and simulated annealing (SA). This hybrid approach aims to achieve a critical balance between efficiently searching a vast solution space (exploration) and meticulously refining promising regions within that space (exploitation), ultimately leading to superior ransomware detection capabilities.

JAYA, a powerful optimization algorithm, excels in exploring the solution space, but it can sometimes struggle with local optima traps. To address this limitation, we integrate SA, an algorithm renowned for its refined local search capabilities. This strategic combination allows SMO-BJAYA-SA-SVM to effectively navigate the solution space, avoiding getting stuck in suboptimal solutions and efficiently converging towards the optimal one.

Through rigorous empirical evaluation, we have demonstrably proven the effectiveness of our proposed approach. Compared to existing algorithms, SMO-BJAYA-SA-SVM achieves superior performance, boasting an accuracy rate. In essence, this research presents a unique hybrid system that merges the strengths of various algorithms to create a robust and efficient ransomware detection solution. This approach paves the way for enhanced security on portable devices, protecting users from the ever-evolving threat of ransomware attacks.

Our Contributions

Our key contributions consist of the following:

1. Provides a taxonomy for the most current research projects that are trying to improve machine learning techniques for ransomware detection.
2. Makes use of an extremely difficult, imbalanced dataset that is intended to mimic the present-day situation of the Android marketplace, which is marked by a notably small percentage of the Ransomware class relative to the normal app class.
3. Provides a novel hybrid detection technique that makes use of binary JAYA along with SA to properly balance the exploration and exploitation search milestones.
4. Employs the SMOTE oversampling approach to rectify the imbalanced data while maintaining the original information integrity.
5. Presents a novel method that, through the use of a unified framework, simultaneously optimizes feature selection, the number of k-nearest neighbors, the imbalance ratio of the SMOTE, and the cost parameter of the SVM. This method effectively addresses multiple challenges in classification tasks.
6. Examines how well four evolutionary selection algorithms-Random, Roulette wheel, Linear rank, and Tournament-work when it comes to ransomware detection.
7. The empirical results show that the proposed SMO-BJAYA-SA-SVM approach is superior to other algorithms. The accuracy rate, precision, recall, and F1 score of the suggested approach were all 98.7%, 98.6%, and 98.7%, respectively.

Paper Organization

The current paper is structured as follows: "Related Work" section presents an extensive review of the latest literature on ransomware detection, highlighting the significant research conducted in this field. Section "Algorithms" provides a detailed explanation of our proposed approach, including the algorithms used for detecting ransomware, the dataset utilized, and the evaluation measures employed. In "Results and Discussions' section, we discuss the results of our study, outlining the critical features necessary for effective ransomware detection and providing a detailed comparison with other algorithms. Lastly, "Conclusion" section offers concluding remarks, summarizing the main contributions of our research and suggesting potential areas for future research.

Related Work

Ransomware has become one of the most prevalent and devastating types of cyber-attacks, where the attacker takes control of the victim's system or data and demands a ransom to restore access. Traditional signature-based methods are not effective in detecting new or unknown ransomware variants, leading to the need for advanced methods such as machine learning and deep learning to detect and mitigate the ransomware threat.

Several research studies employed machine learning algorithms to detect and classify mobile malware. Wang et al. [11] analyzed network traffic data by extracting patterns, data streams, and protocols from the network. The extracted data was then used for the training of machine learning model, which resulted in a 97.5% detection rate and a false positive level of less than 0.5%. They compared their approach

with respect to other methods such as permission and code analysis and found it to be superior in both accuracy and false positives. Another similar research by Bae et al. [12] investigated the use of machine learning algorithms such as decision trees, support vector machines, and neural networks in the detection of ransomware threats. The highest accuracy was reached by neural networks at 98.94%, support vector machines came second with a percentage of 97.66%, and decision trees came third at 94.68%. This work highlights the potential of machine learning for ransomware detection, which may be improved using more advanced techniques.

The performance of six supervised machine learning algorithms was assessed in the study conducted by Manzano et al. [13]. They used CICIDS2017 dataset, which comprises diverse network traffic such as usual traffic and traffic generated by various kinds of ransomware. The data is cleaned, and 78 attributes are derived per network flow. They later used six various supervised machine learning algorithms to train models on the data and test their accuracy and execution time in the following check. They have found that all six algorithms have high accuracy performance with F1 scores being in a range from 0.984 to 0.997. Nevertheless, the execution time varies considerably, as some algorithms may require several hours to train on the whole dataset. The authors observed that random forest is the most successful in terms of accuracy and reasonably fast speed.

The authors in Soi et al. [14] offer a detailed analysis of the detection of Android malware, covering the urgent requirement of both accuracy and interpretability in machine learning models. Their novel method uses static analysis to extract features that are meaningful, in particular, APIs from the DEX Call Graph, delivering an intelligible and prompt justification for the model's decision. However, by shifting emphasis to the most critical API calls and using NLP-based approaches like TF-IDF and Word2Vec embedding, the proposed methodology provides similar accuracy to the current models, but increases interpretability. The accuracy of the CNN-based classifier utilizing a dataset of more than 48,000 samples representing the years from 2008 to 2022 is 87.3% while the F1 score is also 87.3%. This highlights the system's ability to identify both benign and malicious applications. This study adds to the progress in the detection of Android malware and also reveals core behaviors intrinsic to these applications to help with the prevention of possible threats in mobile device security.

The authors in Kim et al. [15] explored a deep learning method for detecting mobile ransomware. They extracted various features from over 41,000 app samples, analyzing elements like manifest files, dex files, and .so files within APKs. These features aimed to capture different aspects of the applications, ultimately enriching the information and better identifying ransomware threats. Their approach achieved a reported detection rate of 98%. Masum et al. [16] employed various classifiers on ransomware detection and presented a framework for selecting features. The achieved results showed that random forest achieved the best accuracy results. Zhang et al. [17] proposed a static method to classify ransomware. N-grams were generated from the opcode from ransomware samples. The Term Frequency-Inverse Document Frequency (TF-IDF) was computed for each N-gram to select feature N-grams. The TF values of the feature N-grams were the input for different machine learning methods to execute ransomware classification. The experiments conducted using real datasets showed that the proposed method achieved the best accuracy of 91.43%.

Abdullah et al. [18] propose a novel approach for detecting Android ransomware using dynamic features. The authors argue that static features, which are commonly used in existing ransomware detection methods, have limitations in identifying sophisticated ransomware variants. The proposed approach collects dynamic features from Android applications during runtime, which include system calls, API invocations, and network traffic with the aim of classifying them as either benign or ransomware. The authors conducted experiments on a dataset of over 2000 Android applications, including ten ransomware families, and achieved a detection rate of 98.5%. The results demonstrate high detection rates, which suggest that the proposed method has potential for practical applications in mobile security.

Gera et al. [19] tested the effectiveness of ransomware on Android mobile devices and revealed its serious impact on device functionality and user safety. They aimed to deliver a detailed analysis of the effects of ransomware attacks on the Android ecosystem with a view to how ransomware threats are changing for mobile users in the digital world of today. The authors conducted experiments involving five Android security systems: These are Google Play Protect, Kaspersky Internet Security, Norton Mobile Security, Avast Mobile Security, and Bitdefender Mobile Security. The results highlight the urgency of developing potent security tools and monitoring for changing cyber threats in the mobile environment, therefore, focusing on the importance of constant updates and improvements of defensive mechanisms to safeguard devices against the detrimental effects of ransomware and other versions of malware.

Bibi et al. [20] offer an advanced method of android ransomware detection, which includes static and dynamic analysis, multiple feature filtration as well as recurrent neural networks (RNNs). Their approach decodes static code attributes and real-time runtime behaviors, allowing the selected features to be fed into the RNN model. Evaluations on a dataset of 8700 samples demonstrate high detection accuracy (99.1%) and a low false positive rate (0.52%), outperforming other methods. These results suggest the proposed method's effectiveness in detecting Android ransomware and its potential real-world application. Abbasi

et al. [21] employed PSO in detecting and classifying ransomware and discussed the importance of the features for the groups. Alzubi et al. [22] suggested an approach of ransomware detection oriented to SVM and Harris hawks optimization (HHO). HHO was the function to tune the hyperparameters of the SVM classifier in so doing SVM performed the classification task. The experiments used CICMalAnal2017 dataset to test the algorithm. The results showed the ability of the HHO-SVM method to measure the features' importance and to assess the relationship between the detected malware and the features.

The authors in [23] provided Android ransomware detection using both machine learning (ML) and deep learning (DL) technologies. They evaluated the effectiveness of different ML and DL models, such as decision trees, support vector machines, k-nearest neighbors, feedforward neural networks, and tabular attention networks, through rigorous experimentation using an open-source dataset, which contained 392,035 records with 85 attributes. They highlighted important traffic characteristics that influence detection performance and proposed an ensemble technique that combines strengths of multiple algorithms to improve resistance to emerging ransomware threats. Many pre-processes were applied such as binarizing labels, undersampling for handling class imbalance, and converting categorical variables to the numerical ones. Feature scaling methods were used to solve problems like high training loss, confirming good learning of the data from the but model. To select the most significant features for classification, some feature selection techniques like forward feature selection and feature importance were used, and consequently, 19 features were finalized. They evaluated their approach using several supervised learning models such as decision trees (DT), support vector machines (SVM), k-nearest neighbors (KNN) and the ensemble techniques, and deep learning models like feedforward neural networks (FNN) and tabular attention networks (TabNet). DT outperformed the other classifiers, with an accuracy of 97.24%, precision of 98.50%, and F1 score of 98.45%, whereas, in terms of the highest recall, SVM achieved 100%.

The authors in [24] present a deep analysis of Android malware detection with particular emphasis on the effectiveness of image-based analysis techniques. They suggest a new paradigm that integrates visual features obtained from AndroidManifest.xml with DEX file's data section. Employing a temporal convolution network (TCN), they obtained considerable improvements in the accuracy of the detection and reached an astonishing precision of 95.45% and recall rate of 95.45% with a total detection accuracy of 95.44%. Comparing with traditional convolutional neural network (CNN) and lightweighted MobileNetV2 models, through wide experiments, the TCN-based method performs better in using features from bytecode image sequence and improves Android malware detection efficiency. The research emphasizes the crucial role of effective malware detection techniques, particularly in the context of Android ecosystem growth that existing approaches such as Google Play Protect have shown limitations.

The authors of Yadav et al. [25] proposed an automated Android malware detection method. Their paper presented a deep learning (DL) approach leveraging 26 state-of-the-art pre-trained convolutional neural network (CNN) models combined with support vector machine (SVM) and random forest (RF) classifiers for big data learning and stacking. The EfficientNet-B4 model, based on CNNs, exhibited remarkable results achieving an impressive 95.7% accuracy in binary classification of Android malware images from the dataset, surpassing all other models. This study emphasized the pressing need for efficient malware detection systems due to the exploding rate of malware, particularly considering Android's dominant market share and the increased vulnerability caused by fragmentation and the surge of mobile banking Trojans during the COVID-19 pandemic. Utilizing image-based representations of Android malware, the approach converted DEX files of over 2 million benign and malicious Android apps into RGB color images. This not only simplified the detection process but also demonstrated robustness against code obfuscation.

The author in [3] provided a detailed description of the serious security issue that is posed by ransomware to mobile devices. The author discussed the shortcomings of present anti-malware solutions that cannot effectively cope with evolving ransomware variants that use sophisticated evasion techniques. The author has used a similar approach to ours, adopting the Binary JAYA Optimization Algorithm (BJAYA) for ransomware detection in mobile devices. The performance of BJAYA is validated against a range of datasets such as 0-1 knapsack instances and real ransomware data, and it is evident that BJAYA outperforms the other algorithms in terms of detection rates and specificity. Especially, BJAYA beat other algorithms in 85% of 0-1 knapsack datasets and showed quite high sensitivity and Gmean value of 97% and 98.2%, respectively, in detecting ransomware. These findings emphasize the importance of employing optimization algorithms such as BJAYA in alleviating the growing danger of ransomware attacks on mobile devices.

In [26], the authors combined the Grey Wolf Optimizer (GWO) with an ensemble of SVM for ransomware detection to optimize the number of clusters. Also, it compared the results by applying the SVM single classification. The results showed that SVM-GWO outperformed the corresponding SVM classifier using the accuracy measure. Almomani et al. [27] proposed a hybrid evolutionary approach for detecting Android ransomware in the context of highly imbalanced data. The approach combined a feature selection technique based on PSO with a classifier ensemble composed of decision trees and k-nearest neighbors (KNN). The experimental

results showed that the proposed approach outperformed other state-of-the-art methods in terms of accuracy, precision, and recall, achieving an accuracy rate of 99.64% and a false positive rate of 0.03%. The hybrid approach was effective in addressing the challenges of imbalanced data in Android ransomware detection, where the number of normal instances was significantly higher than the number of ransomware instances. The authors in [28] combined the Salp Swarm Algorithm (SSA) and kernel extreme learning machine (KELM). The SSA was used to optimize the hyperparameters of the KELM to classify ransomware. The SSA-KELM achieved an area under the curve of a value of 98%.

The related works emphasize the strategic role machine learning techniques play in tackling the growing threat posed by mobile ransomware. Researchers now have at their disposal a wide variety of technology solutions based on machine learning, some of which are targeted at addressing the changing threat landscape. Machine learning entails a range of methods including feature extraction, selection, and classification that have been proven to be sound in discriminating ransomware from benign software. From static feature-based deep learning-based approach to the feature collection during runtime, each method has individual strengths and abilities.

Despite the successes of these techniques in accurately detecting real-world mobile ransomware samples, there is a wealth of optimization algorithms that remain untapped, which could enhance the classification results of the integrated learning algorithms. In addition, as the existing research revolutionized the current technologies in mobile ransomware detection, there is still room for further improvement. One promising direction which has been explored is the usage of optimization algorithms such as the Binary JAYA Optimization Algorithm (BJAYA) proposed in the paper. With the use of these optimization techniques, the researchers can fine-tune and optimize the performance of machine learning algorithms to achieve higher accuracy and greater robustness in detecting and countering ransomware attacks on mobile devices.

Therefore, with the evolution of the threat landscape and adversaries using more advanced strategies, the need for continuous innovative and adaptive ways of detecting ransomware becomes ever more important. Optimization techniques such as JAYA, when incorporated into the machine learning toolkit, represent a preventive measure that will help in containing ransomware attacks and keeping the mobile ecosystems secure. Researchers can be more successful in the development of machine learning-based strategies by using this untapped potential thereby augmenting the ability of such approaches to combat the mobile ransomware thus ultimately ensuring the confidentiality and integrity of mobile devices and user data.

The proliferation of mobile ransomware presented an increasing threat, and machine learning techniques emerged as promising results for detecting and mitigating such threats. Various machine learning-based approaches for mobile ransomware detection were proposed in the literature, leveraging feature extraction, selection, and classification, as well as hybrid methods that combined machine learning with behavior-based analysis. Despite the successes of these techniques in accurately detecting real-world mobile ransomware samples, there is a wealth of optimization algorithms that remain untapped, which could enhance the classification results of the integrated learning algorithms. In this paper, we proposed the use of the JAYA algorithm for ransomware detection, with multiple modifications to improve its performance.

Methodology

This section delves into the algorithms, methodology, data, and evaluation metrics of our research on ransomware detection. It emphasizes the crucial role of accurate detection in mitigating ransomware attacks. We thoroughly explain the employed algorithms, highlighting the critical features for effective identification. Our proposed methodology for detecting ransomware, drawing from a real-world dataset with malicious and benign samples, is meticulously described. This real-world focus underscores the practical relevance of our research in tackling genuine cybersecurity challenges. Furthermore, we outline the evaluation metrics used to assess the effectiveness of our approach, emphasizing the importance of appropriate metrics for reliable and valid results. This comprehensive overview of our methods and data highlights the significance of our findings for advancing cybersecurity.

Algorithms

Simulated Annealing Algorithm

Kirkpatrick's 1983 work introduced the SA algorithm, a trajectory-based approach that starts optimization with a single solution and iteratively refines it through local searches [8]. The refinement of the current solution is performed by searching the neighboring solutions. At each iteration, a decision is taken depending on the computed value that subtracts the objective values of the neighboring individual and the best individual. If the resulting value is less than 0, then the neighboring individual is announced as the best individual. If a random value is less than the Boltzmann probability, then the SA accepts the worse neighbor individual. The Boltz-

mann probability function $P = e - \Theta \times Temp$, where Θ is the parameter that computes the difference between the fitness of the current individual and the best individual. At the same time, Temp is the temperature parameter that decreases throughout the optimization process. If no condition is satisfied, then the best individual is not updated. Temp parameter is updated at the end of each cycle. See Algorithm 1.

Algorithm 1 Simulated annealing (SA)

Input:$Temp$, $Fitness$, CI
Output: $Optimal\_individual$, $Optimal\_cost$
$Fitness$ is the objective function
$Temp$ is the temperature
CI = Current individual
NI the neighboring individual
$Optimal\_individual = CI$
$Optimal\_cost = Fitness(CI)$
Initialize $Temp$
$Curr\_individual = CI$
$Curr\_cost = Fitness(CI)$

while (Stopping criterion is not satisfied) **do**
NI = Produce new neighboring individual of CI

$New\_cost = Fitness(NI)$

$th = New\_cost$ - $Curr\_cost$
if $(th < 0)$ **then**
$Optimal\_individual = NI$
$Optimal\_cost = New\_cost$
$Curr\_individual = NI$
$Curr\_cost = New\_cost$
else if $(random(0, 1) \leq e^{\frac{-th}{Temp}})$ **then**
$Curr\_individual = NI$
$Curr\_cost = New\_cost$
end if
$Temp = \alpha\ Temp$
end while

JAYA Algorithm

The JAYA algorithm [7] is a multi-solution algorithm. This means that it starts the optimization process with a set of solutions that will be updated by changing their positions in the search space to approach near the optimal solution and to avoid the positions of the worse solutions. This concept is named the "survival of the fittest." The main merits of the JAYA algorithm that encouraged us to use it in this paper are the few parameters that need to be tuned. It uses only the maximum iterations and the size of the population parameters, but there are no specific algorithmic parameters to experiment with. The steps of the JAYA algorithm can be summarized as follows:

- The parameters of the algorithm such as the number of individuals and the maximum number of iterations are initialized. Furthermore, the objective function and the solution representation have to be identified.
- Generating the initial random population. The population has N solutions with D dimensions for each solution $S_i = (S_{i,1}, S_{i,2}, \ldots, S_{i,d})$. Equation (1) is used to generate the random solutions.

$$S_{i,j} = S_j^{lb} + (S_j^{ub} - S_j^{lb}) \times U(0, 1) \tag{1}$$

  where $i \in (1, 2, \ldots, N)$, and $j \in (1, 2, \ldots, d)$. $S_{i,j}$ is the j_{th} element in the i_{th} solution. S_j^{lb}, S_j^{ub} are the minimum and maximum values of the element in the j position. $U(0, 1)$ is a function that is used to generate uniform random values distributed in (0,1). Later on, the objective value of the generated random solutions is computed using the objective function $Fitness(S)$.
- Identify the best and the worst solutions in the swarm. $Fitness(S_{best}) \leq Fitness(S_i)$, $i = 1, 2, ..., N$. $Fitness(S_{worst}) > Fitness(S_i)$, $i = 1, 2, ..., N$.
- The JAYA operator is applied to randomly modify the elements of each solution at each iteration as in Eq. (2):

$$\begin{aligned}\overline{S}_{i,j} = S_{i,j} + random_1 \times (S_{best,j} - |S_{i,j}|) \\ - random_2 \times (S_{worst,j} - |S_{i,j}|)\end{aligned} \tag{2}$$

  where $\overline{S}_i$ and S_i are the new and the current solutions, respectively. $\overline{S}_{i,j}$ is the modified value of the element value $S_{i,j}$. $random_1$ and $random_2$ are two random numbers $\in (0, 1)$ to explore the search space. The term $random_1 \times (S_{best,j} - |S_{i,j}|)$ shows how the current solution moves closer to the best solution $S_{best,j}$. On the other hand, the term $-random_2 \times (S_{worst,j} - |S_{i,j}|)$ shows how the current solution avoids the worst solution $S_{worst,j}$. The absolute values $|S_{i,j}|$ in Eq. (2) are subtracted from $S_{best,j}$ and $S_{worst,j}$ to improve the exploration of the JAYA algorithm.
- When the new solution $\overline{S}_i$ is generated, it compares to the current one S_i stored in the i_{th} location in the swarm. If the objective value of $\overline{S}_i$ is better than the objective value of Si, then the new solution replaces the current one in the swarm
- The previous three steps are repeated until a stopping criterion is met such as the maximum number of iterations is reached.

SMOTE Oversampling Method

Synthetic minority oversampling (SMOTE) is an algorithmic solution method to address the problem of imbalanced data. The method relies on generating artificial instances that are similar to the actual ones [29]. For example, SMOTE redistributes the Mj and Mn classes where Mj is a dominant

class, and Mn is a rare in the imbalanced dataset D with N instances in such a way to make more balance between them. It synthesizes new interpolated instances from the data instances to increase the ratio of the Mn class.

Evolutionary Selection Schemes

Survival of the fittest is the standard selection concept that is applied in the JAYA algorithm. This may reduce the diversity of the algorithm and bias towards the best solution [30]. Therefore, this study applies other evolutionary selection schemes [31]:

- Random: All solutions get the same selection probability to select the value of any decision variable.
- Linear rank: It determines the selection probability of the solutions using the individuals' rank in place of their fitness. The highest rank is given to the best solution, whereas the lowest rank is given to the worst solution.
- Global-best: The best solution has a unity probability value to be selected. On the other hand, the remaining solutions have zero selection probability.
- Roulette wheel: The probability of selecting a solution depends on its fitness value, so it is proportional to the fitness of a solution. This means that a solution with a higher fitness has a larger probability of being selected.
- Tournament: It gets a set of individuals from the entire swarm. The number of selected solutions is determined by the tournament size. These solutions are selected randomly, and then a solution with the best fitness value is selected from a tournament.

Proposed Hybrid JAYA-SA

The proposed ransomware detection method is based on applying a wrapper-based feature selection framework. This framework applies a hybrid evolutionary search model which is composed of JAYA population-based and SA trajectory-based algorithms. The integration of the SA algorithm in the structure of JAYA improves the exploitation power of the standard JAYA. The SA is implemented at the end of each cycle of the JAYA algorithm to search for a better individual than the optimal individual generated by the JAYA algorithm. The optimal individual gained by the JAYA algorithm at the end of each cycle is received by the SA algorithm as an initial individual. This individual will be refined by the optimization process of SA. If SA finds a better individual, it will replace the current optimal individual with the new one. Therefore, the hybrid model generated by integrating SA with JAYA will enhance the local search power of the JAYA and find better individuals.

A SMOTE oversampling method is used for tackling the imbalanced class problem by increasing the ransomware instances and balancing the ransomware and benign classes for better performance. The SVM is used as a classification algorithm to generate the training model that performs the classification task for applications as malicious and benign.

Figure 1 shows the proposed ransomware approach, which is called SMO-BJAYA-SA-SVM. The BJAYA-SA algorithm optimizes simultaneously the number of selected features, the K nearest neighbors parameter of the SVM algorithm, and the sampling ratio of the SMOTE method. A geometric mean-based fitness function is used to evaluate the classification model produced by the optimization process to improve it in later iterations.

The JAYA algorithm is a continuous optimization algorithm. However, adapting it to deal with the feature selection problem which is a discrete optimization problem needs some modifications. In this study, a sigmoid transfer function which is given in Eq. (3) is applied to generate binary solutions.

$$Sigmoid(S) = \frac{1}{1 + e^{-x}} \tag{3}$$

Each solution has three parts which are as follows:

1. Feature selection part: These features are selected based on a list of n binary values shown as the first part of the solution. Each binary value in the first part indicates if a feature is selected or not. For example "0" means the feature is not chosen while "1" means that the feature is chosen. Therefore, the number of 1's in the first part indicates the number of selected features.
2. SMOTE part: In this part, the number of neighbors (K) and the oversampling ratio are represented. For SMOTE, the number of neighbors indicates how many neighbors are used to synthesize the malicious application rare class. Four binary digits (0000–1111) reflect a decimal range (0–15), respectively. Furthermore, the oversampling ratio (r) indicates the number of oversampled examples of the malicious class divided by the number of examples of the normal class. It consists of six binary digits (000000–111111) of a range (0.0 and 0.63), respectively.
3. The SVM part: The third part concerns the cost coefficient parameter of the SVM classifier. It allocates eight binary digits (00000000 to 11111111) that corresponds to a float value in the range (0.0 and 2.55). The cost parameter gives indication on how the generated model fits the training data where a high cost indicates low fit while a low cost indicates high fit. This can help in avoiding the overfitting in the training phase.

A fitness function is used internally to decide the goodness of each solution in the proposed SMO-BJAYA-SA-SVM.

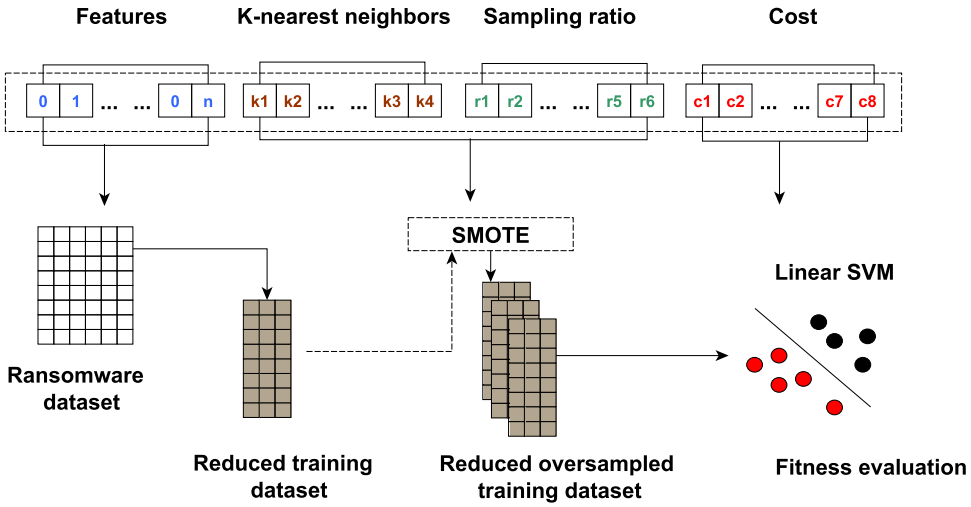

Fig. 1 The structure of the solution in the proposed ransomware detection method

This can indicate its ability to guide the optimization process towards optimality and alleviate the stagnation in local minima. In this study, the main objective is to maximize the performance of the classification algorithm. The fitness is formulated using Eq. (4), which is 1 minus the Gmean of classification. The Gmean is identified by Eq. (7).

$$Fitness = 1 - \sqrt{Gmean} \tag{4}$$

The steps of the proposed SMO-BJAYA-SA-SVM can be summarized as follows:

1. Initializing solutions: A set of solutions is randomly initialized, where each individual is composed of three parts (feature selection, K, sampling ratio, C) as shown in Fig. 2.
2. Updating solutions: Reposition the solutions through the iterations to discover more locations to approach a promising place near the optimal solution.
3. Mapping solutions: Extracting values from the solutions to select features, and as an input to determine the parameters of the SMOTE method and the SVM classifier.
4. Evaluating the fitness of solutions: The Gmean is applied to evaluate the improved solutions of each iteration based on Eq. (4).
5. Stopping criterion: Terminates the procedure when a the maximum number of iterations is satisfied.
6. Testing: Evaluate the classification model produced by the SMO-BJAYA-SA-SVM algorithm on the testing part using different evaluation measures. Figure 2 shows the steps by which SMO-BJAYA-SA-SVM follows.

Algorithm 2 JAYASA algorithm

Initialize the parameters: $population\_size(N)$, and $Max\_iterations$
Initialize the solutions $(S_{i,j})$
Compute $Fitness(S_i)$, $i = 1, 2, ..., N$
while $iteration \leq Max\_iterations$ **do**
Identify the best solution (S_{best})

Identify the worst solution (S_{worst})

for i = 1: N **do**
for j = 1: d **do**
Set $random_1 \in [0, 1]$

Set $random_2 \in [0, 1]$

$\overline{S}_{i,j} = S_{i,j} + random_1 \times (S_{best,j} - |S_{i,j}|) - random_2 \times (S_{worst,j} - |S_{i,j}|)$
end for

if $Fitness(\overline{S}_i) \leq Fitness(S_i)$ **then**
$\overline{S}_i \leq S_i$
end if

end for

Apply SA
$iteration = iteration + 1$;
end while

Ransomware Dataset

The ransomware dataset used in this study is imbalanced since there is a large difference between the malicious ransomware class (rare class) and the normal or benign class (majority class). Using numbers, there are 500 applications classified as ransomware while there are 10,153 applications that are classified as benign. Four steps have been applied to create this dataset in the Security Engineering Lab (SEL)[2]:

- Data collection: The Android applications have been collected from different market repositories as packages (APK). The malicious applications were downloaded

[2] https://sel.psu.edu.sa/Research/datasets/2020_RansIm-DS.php.

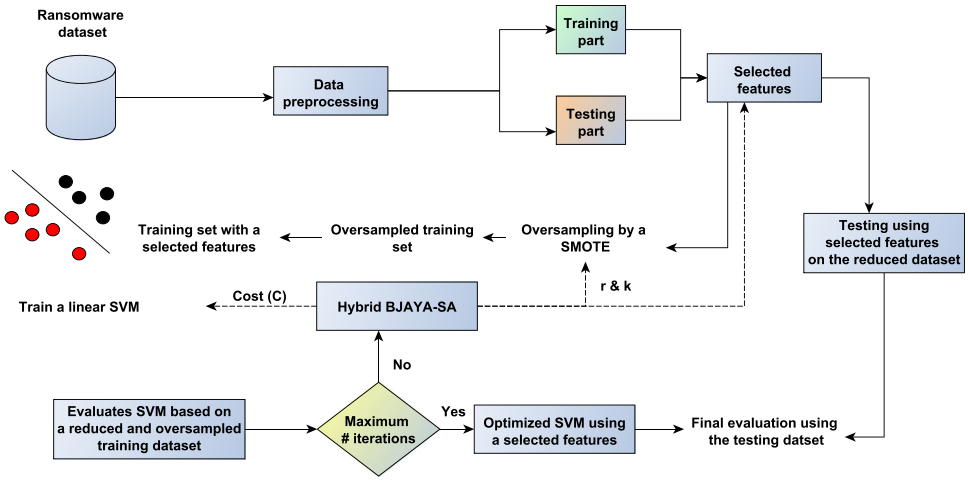

Fig. 2 Flowchart of the proposed SMO-BJAYA-SA-SVM

from different online malware platforms such as VirusTotal, RansomProper, and koodous whereas Google Play store was used to download normal applications.

- Decomposition: The process of extracting some necessary files from the APK file such as manifest file and .smali files. These files contain the metadata and application permissions.
- Parsing: Extracting the features of the application from the manifest file and .smali files. The number of features in this dataset is 389 features.

Evaluation Measurement

The confusion matrix is the main source to derive these measures. Table 1 shows the confusion matrix where the positive class is the malicious class and the negative class is the benign class. The TP indicates the instances that are malicious and are detected as malicious. The TN indicates the instances that are detected as benign and are actually benign, whereas the FP indicates the instances that are detected as malicious but are actually benign, and the FN are instances that are detected as benign but are malicious.

The sensitivity describes the ability of the learner to detect the malicious applications as in Eq. (5), specificity describes the ability of the learner to detect the benign applications as in Eq. (6), and the Gmean describes the balance of performances detection for the malicious and benign classes as in Eq. (7).

Table 1 Confusion matrix

	Actual malicious	Actual benign
Detected malicious	TP	FP
Detected benign	FN	TN

$$Sensitivity = \frac{TP}{TP + FN} \tag{5}$$

$$Specificity = \frac{TN}{TN + FP} \tag{6}$$

$$Gmean = \sqrt{\frac{TP}{TP + FN} \times \frac{TN}{TN + FP}} \tag{7}$$

Results and Discussions

The first experiment investigates the effect of different selection schemes on the performance of the proposed method to detect ransomware when the benign/malicious class distribution is imbalanced. Five variants of the proposed method are investigated by using five different evolutionary selection schemes: Random, Linear rank, Global-best, Roulette, and Tournament. Each method is embedded in the proposed SMO-BJAYA-SA method and investigated using different scenarios of the swarm size and number of iterations parameters. The used evaluation measures sensitivity, specificity, and Gmean are shown in Table 2. Furthermore, it shows the optimized values achieved for the SVM cost and the number of selected feature parameters.

Table 2 observes that the Roulette wheel achieves the highest results compared with the results obtained by the other selection schemes. For example, when the swarm size is 50 and the number of iteration is 50, the Roulette wheel obtained the best sensitivity, specificity, Gmean, and the minimum number of features by having 97.5%, 99.8%, 98.6%, and 171, respectively. Yet, it obtained the best specificity

Table 2 The results of the proposed method using different selection schemes with different parameter scenarios. The best results appear in parenthesis

Selection scheme	Swarm size	#iterations	Sensitivity	Specifity	Gmean	Cost	#features
Random	25	25	0.945	0.986	0.965	0.51	(181)
	50	50	0.945	0.992	0.968	1.32	187
	100	100	0.956	0.991	(0.974)	2.00	194
Roulette wheel	25	25	0.950	(0.994)	0.972	0.65	198
	50	50	(0.975)	(0.998)	(0.986)	1.01	(171)
	100	100	0.926	(0.995)	0.960	0.86	200
Global-best	25	25	0.963	0.987	0.975	0.54	197
	50	50	0.956	0.992	0.974	0.55	183
	100	100	0.950	0.985	0.968	1.02	209
Linear rank	25	25	0.969	0.971	0.970	0.34	190
	50	50	0.945	0.997	0.970	2.32	208
	100	100	0.945	0.989	0.966	1.12	(176)
Tournament	25	25	(0.981)	0.981	(0.981)	2.21	186
	50	50	0.963	0.970	0.966	1.34	215
	100	100	(0.969)	0.975	0.972	1.01	187

among all when the swarm size was 25 and 100, having 99.4% and 99.5%, respectively. The Tournament can obtain the best results using the sensitivity and Gmean when the swarm size is 25 by having 98.1% and using the sensitivity when the population size is 100, gaining 96.9%. Furthermore, Random achieved the topmost Gmean when the swarm size is 100 (97.4%) and the minimum number of features (181) at a swarm size of 25. The Roulette wheel scheme outperformed them mainly when the swarm size is 50, and the number of iterations is 50, where the best cost parameter also was 1.01.

Figures 3, 4, and 5 show the convergence curves of the JAYA algorithm using different selection schemes using different parameters settings of the swarm size, and the number of iterations (25, 25), (50, 50), and (100, 100). The figures show the ability of the algorithm to reach the optimal fitness value during the training phase across iterations. The figures show that the proposed algorithm has obtained the best convergence behavior towards the best fitness when the Roulette wheel selection scheme was applied. On the other hand, the Random selection scheme has the poorest convergence trend. Also, for all the selection schemes, the algorithm gradually converges towards the optimal value, but this was affected by the number of solutions and number of iteration parameters. The best convergence behavior was obtained with (100, 100) parameter setting for the swarm size and number of iterations. Therefore, for the subsequent experiments, the SMO-BJAYA-SA will be applied with the Roulette wheel selection scheme and with (100,100) parameter setting.

Table 5 shows the results of running time for the proposed SMO-BJAYA-SA-SVM versus other classification algorithms and other metaheuristic algorithms. It can be seen that the classification algorithms have close values and that minimum running time was achieved by a random forest classifier. The metaheuristic algorithms achieved close running time, but it is larger than the time consumed by the standard classifiers. However, metaheuristic algorithms try to get the best subset of features with a smaller size and maximum performance within a reasonable running time instead of exponential running time that results from producing all the combinations of features. The high-quality results are achieved in Table 2 which explains the larger time compared to the standard classifiers.

Comparison with Standard Classification Algorithms

The best version of SMO-BJAYA-SA-SVM, which uses the Roulette wheel selection scheme, will be compared with other commonly used classifiers including the Naive Bayes, multilayer perceptron, random forests, and decision tree. For the multilayer perceptron algorithm, the number of hidden neurons is set to 10, and for the random forests, 100 trees are used [26]. The results are shown in Table 3. The SMO-BJAYA-SA-SVM achieved the best Gmean value with a value of 98.6%, even that random forest obtained a close result (97.0%). Regarding the sensitivity, the SMO-BJAYA-SA-SVM also achieved the best value of 97.5%, while the decision tree, and the multilayer perceptron obtained the worst results by having 93.6% and 93.7% respectively. Using specificity, the multilayer perceptron achieved the best result of 99.9%. Although the proposed SMO-BJAYA-SA-SVM method did not achieve the best result using specificity, it achieved a close result which is 99.8%. The proposed SMO-

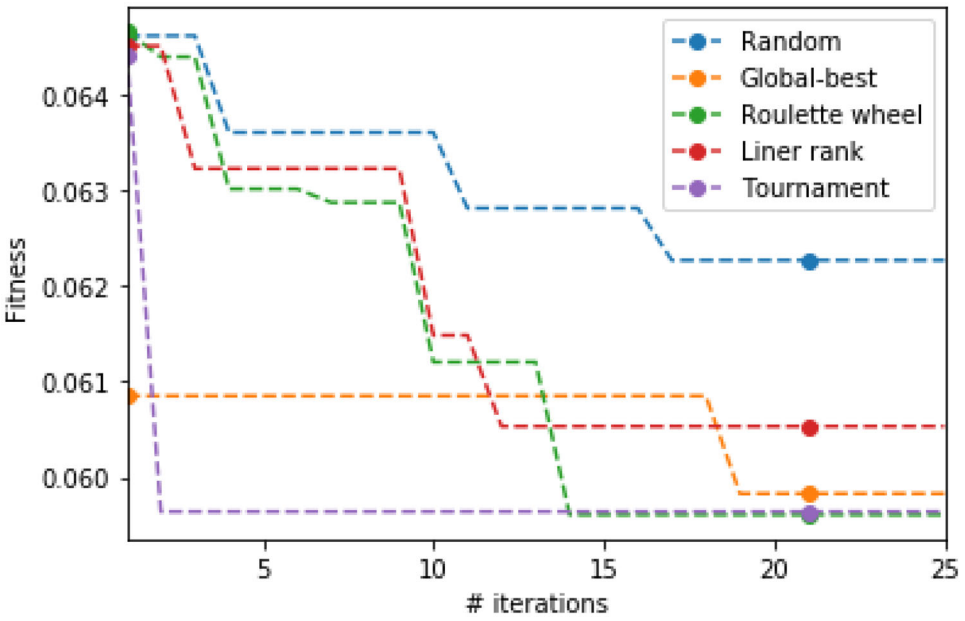

Fig. 3 Convergence scales of different selection schemes for parameter scenario (25, 25)

BJAYA-SA-SVM achieved a better than standard classifiers using sensitivity and Gmean.

Comparison with Other Wrapper-Based Algorithms

Table 4 shows the results achieved by BJAYA for feature selection and using other swarm-based algorithms. These include the Moth Flame Optimization (MFO), Salp Swarm Algorithm (SSA), Grey Wolf Optimizer (GWO), and Particle Swarm Optimization (PSO). For fairness in comparisons, the same settings are adopted having 50 iterations and a population size value of 50. Table 4 shows that the proposed SMO-BJAYA-SA-SVM achieved the results of 99.8%, 97.5%, and 98.6% for the specificity, sensitivity, and Gmean, respectively. These are the best-achieved results compared with other wrapper-based metaheuristic algorithms. Table 5 shows the running time (in seconds) results achieved by BJAYA against other common algorithms and swarm-based algorithms. As seen in Table 5, the standard algorithms have close processing time. However, Naive Bayes took more time than multilayer perceptron and random forest. On the other hand, swarm-based methods have a close running time. However, SMO-BGWO-SA-SVM took more time than other

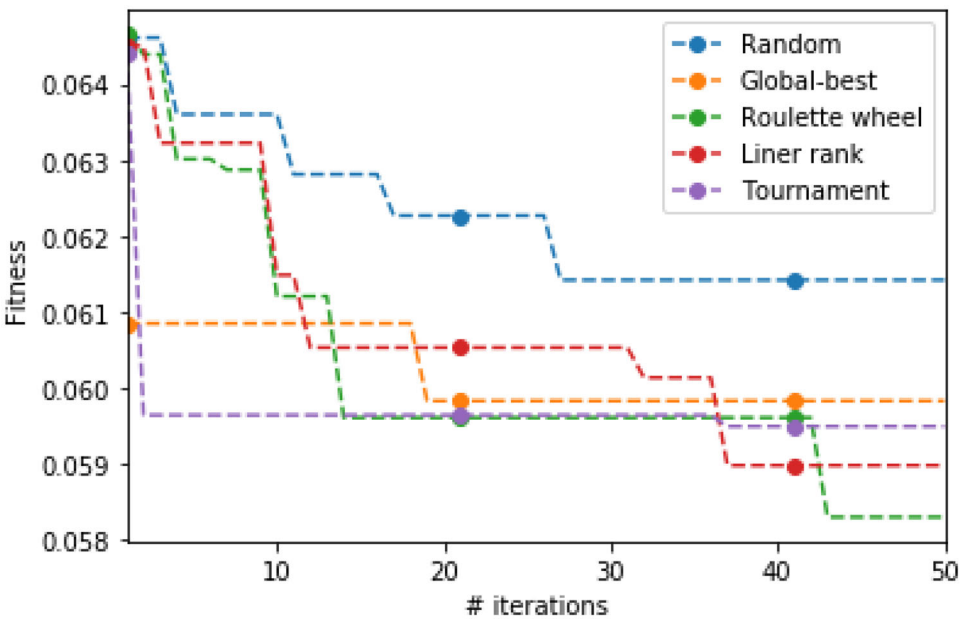

Fig. 4 Convergence scales of different selection schemes for parameter scenario (50, 50)

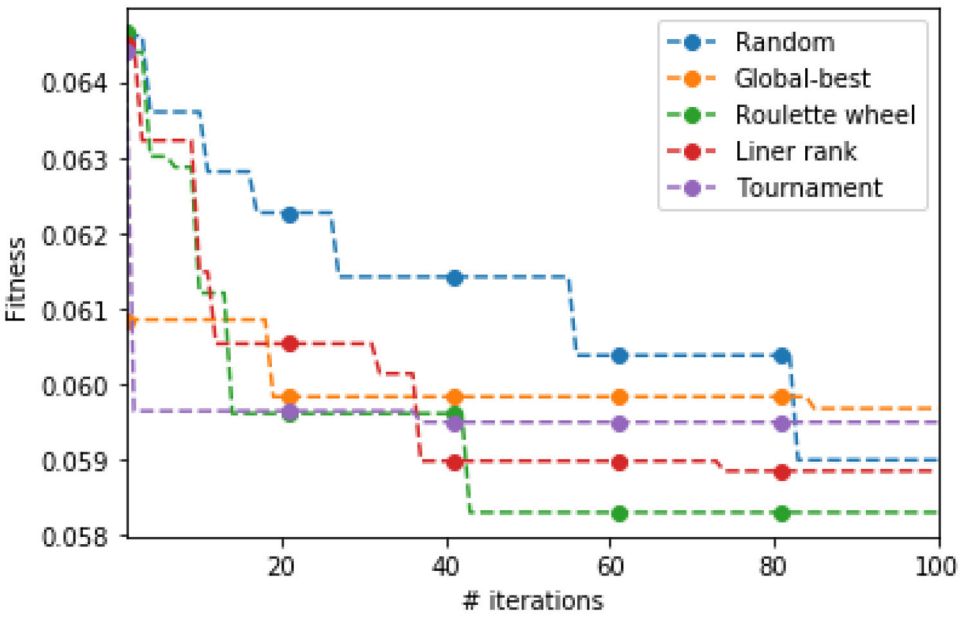

Fig. 5 Convergence scales of different selection schemes for parameter scenario (100, 100)

Table 3 Comparison with common learning algorithms

Standard classifiers	Specificity	Sensitivity	Gmean
Naive Bayes	0.961	0.945	0.959
Multilayer perceptron	(0.999)	0.937	0.962
Random forest	0.976	0.966	0.970
Decision tree	0.914	0.936	0.951
SMO-BJAYA-SA-SVM	0.998	(0.975)	(0.986)

methods. The proposed SMO-BJAYA-SA-SVM took the shortest time to find the best solution among other swarm-based methods. By comparing the standard algorithms and the swarm-based algorithms, it is seen that the swarm-based methods took a longer time to find the best solution. This can be explained that swarm-based methods take a reasonable amount of time to search in the feature space and find the best subset of features. Generating higher-quality results requires more time by swarm-based methods compared with the standard methods.

Table 4 Comparison with other swarm-based algorithms

Standard classifiers	Specifity	Sensitivity	Gmean
SMO-BMFO-SA-SVM	0.988	0.942	0.962
SMO-BSSA-SA-SVM	0.982	0.939	0.957
SMO-BGWO-SA-SVM	0.977	0.958	0.968
SMO-BPSO-SA-SVM	0.989	0.967	0.979
SMO-BJAYA-SA-SVM	(0.998)	(0.975)	(0.986)

Feature Engineering

The proposed detection model identifies the most significant features that are necessary to detect ransomware. This can be known based on the number of times the feature appears in the best individual across 30 independent runs.

An importance score is given for each feature according to the number of times it appears in the best solution over 30 independent experiments. Some of these features are API calls and others are permission features. The permission features have higher scores of more than 80% such as SYSTEM_ALERT_WINDOW, READ_PHONE_STATE, READ_CALL_LOG ACCESS_NETWORK_STATE, KILL_BACKGROUND_PROCESSES. This can be explained that the permission features are exploited by the ransomware to gain access to system resources. As opposed to API calls such as android/system and android/opengl that are mostlyused

Table 5 Comparison with other feature selection methods

Standard classifiers	Time (second)
Naive Bayes	25.61
Multilayer perceptron	21.42
Random forest	20.65
SMO-BMFO-SA-SVM	165.23
SMO-BSSA-SA-SVM	124.58
SMO-BGWO-SA-SVM	198.53
SMO-BPSO-SA-SVM	123.11
SMO-BJAYA-SA-SVM	108.76

by normal apps, many ransomware attack examples rely on blocking access to system resources or displaying alert windows. Moreover, other permissions are exploited to gain information about the phone and network information or disable some running processes such as anti-virus processes. Other permissions such as RESTART_PACKAGES are used to turn off processes in the background. RECEIVE_SMS permission is exploited by ransomware to receive SMS messages to perform the payment online.

Conclusion

In conclusion, the paper presents a novel approach for detecting ransomware using the SMO-BJAYA-SA-SVM algorithm, which combines various optimization and selection schemes to achieve better performance. The proposed method is shown to be effective in tackling the imbalanced class distribution and optimizing the selected features, the SMOTE parameters, and the SVM classifier cost. The results demonstrate that the SMO-BJAYA-SA-SVM algorithm outperforms standard classifiers and other wrapper-based algorithms in detecting ransomware efficiently, with a Gmean of 97.6%. The study also suggests future research directions, such as extending the proposed approach to develop a multi-objective version of the metaheuristic algorithms and adopting deep learning algorithms. The limited availability of ransomware datasets might be assigned to the challenges involved in obtaining and evaluating ransomware samples. These difficulties result from ransomware's complex encryption and obfuscation techniques, which make it difficult to detect and categorize the infection. As a result, the distribution of classes becomes imbalanced, with the non-ransomware class dominating and the ransomware class becoming uncommon. This causes the classifier to favor the dominant class, which causes the overfitting problem and presents extra challenges for SVM during the learning phase. Furthermore, it is challenging to develop security updates that are globally relevant due to the wide variety of software as well as hardware configurations among Android smartphones and tablets, leaving many devices open to ransomware and other forms of assaults. Moreover, substantial losses have been caused by the appearance of new ransomware variations.

Availability of Data and Material All data and material used for preparation are included in the manuscript itself.

Declarations

Competing Interest The authors declare no competing interests.

References

1. Decarolis F, Li M. Regulating online search in the EU: from the android case to the digital markets act and digital services act. Int J Ind Organ. 2023;90.
2. VonderLinden C, Walton J, Melaragno A, Casey W. The visualization of ransomware infection. In 2022 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech). IEEE; 2022: pp. 1–7.
3. Alazab M. Android ransomware detection using binary JAYA optimization algorithm. Expert Syst. 2024;41(1).
4. Boticiu S, Teichmann F. How does one negotiate with ransomware attackers? Int Cybersecur Law Rev. 2024;5(1):55–65.
5. Bashir S, Maqbool F, Khan FH, Abid AS. Hybrid machine learning model for malware analysis in android apps. Pervasive Mob Comput. 2024;97: 101859.
6. Gopinath M, Sethuraman SC. A comprehensive survey on deep learning based malware detection techniques. Computer Science Review. 2023;47: 100529.
7. Zitar RA, Al-Betar MA, Awadallah MA, Doush IA, Assaleh K. An intensive and comprehensive overview of JAYA algorithm, its versions and applications. Arch Comput Methods Eng, 2021; pp. 1–30.
8. Pan X, Xue L, Lu Y, Sun N. Hybrid particle swarm optimization with simulated annealing. Multimed Tools Appl. 2019;78(21):29921–36.
9. Agrawal R, Stokes JW, Selvaraj K, Marinescu M. Attention in recurrent neural networks for ransomware detection. In: ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2019, pp. 3222–3226.
10. Brewer R. Ransomware attacks: detection, prevention and cure. Netw Secur. 2016;2016(9):5–9.
11. Wang S, Chen Z, Yan Q, Yang B, Peng L, Jia Z. A mobile malware detection method using behavior features in network traffic. J Netw Comput Appl. 2019;133:15–25.
12. Bae SI, Lee GB, Im EG. Ransomware detection using machine learning algorithms. Concurr Comput Pract Exp. 2020;32(18):e5422.
13. Manzano C, Meneses C, Leger P. An empirical comparison of supervised algorithms for ransomware identification on network traffic. In 2020 39th International Conference of the Chilean Computer Science Society (SCCC). IEEE, 2020, pp 1–7.
14. Soi D, Sanna A, Maiorca D, Giacinto G. Enhancing android malware detection explainability through function call graph APIs. J Inf Secur Appl. 2024;80.
15. Kim T, Kang B, Rho M, Sezer S, Im EG. A multimodal deep learning method for android malware detection using various features. IEEE Trans Inf Forensics Secur. 2018;14(3):773–88.
16. Masum M, Faruk MJH, Shahriar H, Qian K, Lo D, Adnan MI. Ransomware classification and detection with machine learning algorithms. In 2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC). IEEE, 2022, pp 0316–22.
17. Zhang H, Xiao X, Mercaldo F, Ni S, Martinelli F, Sangaiah AK. Classification of ransomware families with machine learning based on N-gram of opcodes. Futur Gener Comput Syst. 2019;90:211–21.
18. Abdullah Z, Muhadi FW, Saudi MM, Hamid IRA, Foozy CFM. Android ransomware detection based on dynamic obtained features, in: Recent Advances on Soft Computing and Data Mining:

Proceedings of the Fourth International Conference on Soft Computing and Data Mining (SCDM 2020), Melaka, Malaysia, January 22–23, 2020, Springer, 2020, pp. 121–129.
19. Gera T, Singh J, Faruki P, Thakur D. Efficacy of android security mechanisms on ransomware analysis and detection, in: AIP Conference Proceedings, Vol. 2357, AIP Publishing LLC, 2022, p. 040007.
20. Bibi I, Akhunzada A, Malik J, Ahmed G, Raza M. An effective android ransomware detection through multi-factor feature filtration and recurrent neural network. In 2019 UK/China Emerging Technologies (UCET). IEEE, 2019, pp 1–4.
21. Abbasi MS, Al-Sahaf H, Mansoori M, Welch I. Behavior-based ransomware classification: a particle swarm optimisation wrapper-based approach for feature selection. Appl Soft Comput. 2022;108744.
22. Alzubi OA, Alzubi JA, Al-Zoubi A, Hassonah MA, Kose U. An efficient malware detection approach with feature weighting based on Harris hawks optimization. Clust Comput. 2021; pp. 1–19.
23. Albin Ahmed A, Shaahid A, Alnasser F, Alfaddagh S, Binagag S, Alqahtani D. Android ransomware detection using supervised machine learning techniques based on traffic analysis. Sensors. 2023;24(1):189.
24. Zhang W, Luktarhan N, Ding C, Lu B. Android malware detection using TCN with bytecode image. Symmetry. 2021;13(7):1107.
25. Yadav P, Menon N, Ravi V, Vishvanathan S, Pham TD. Efficientnet convolutional neural networks-based android malware detection. Comput Secur. 2022;115: 102622.
26. Qaddoura R, Aljarah I, Faris H, Almomani I. A classification approach based on evolutionary clustering and its application for ransomware detection, in: Evolutionary Data Clustering: Algorithms and Applications, Springer, 2021, pp. 237–248.
27. Almomani I, Qaddoura R, Habib M, Alsoghyer S, Al Khayer A, Aljarah I, Faris H. Android ransomware detection based on a hybrid evolutionary approach in the context of highly imbalanced data. IEEE Access. 2021;9:57674–91.
28. Faris H, Habib M, Almomani I, Eshtay M, Aljarah I. Optimizing extreme learning machines using chains of Salps for efficient android ransomware detection. Appl Sci. 2020;10(11):3706.
29. Wah YB, Ismail A, Azid N, Niswah N, Jaafar J, Aziz IA, Hasan MH, Zain JM. Machine learning and synthetic minority oversampling techniques for imbalanced data: improving machine failure prediction. Comput Mater Cont. 2023;75(3).
30. Khurma RA, Aljarah I, Sharieh A. A simultaneous moth flame optimizer feature selection approach based on Levy flight and selection operators for medical diagnosis. Arab J Sci Eng. 2021;46(9): 8415–40.
31. Shehab M, Khader AT, Al-Betar M. New selection schemes for particle swarm optimization. IEEJ Trans Electron Inf Syst. 2016;136(12):1706–11.