Application of Ant Colony Optimization Improved Clustering Algorithm in Malicious Software Identification

Yong Qian
International School of Technical Education, Sichuan College of Architectural Technology, Deyang, 618000, China

***Abstract*—Due to the increasing threat of malware to computer systems and networks, traditional malware detection and recognition technologies face difficulties and limitations. Therefore, exploring new methods to improve the accuracy and efficiency of malware identification has become an urgent need. This study introduces ant colony algorithm to optimize traditional clustering algorithms and algorithm parameters. The experimental results showed that the improvement rates of the improved algorithm in accuracy, echo value, and false alarm rate were 0.253, 0.115, and 0.056, respectively. The accuracy on the training and validation sets continued to increase and the loss curve continued to decrease. In addition, the improved algorithm had stronger modeling ability for data feature relationships and temporal information. This is of great help in improving the recognition ability of virus and worm software. The improved algorithm had a lower occupancy rate of computing resources compared to other algorithms, but it could also effectively monitor device operation. Compared with traditional methods, this method can more accurately identify malicious software and effectively identify malicious software samples from large-scale datasets. This is of great significance for protecting computer systems and network security.**

Keywords*—Ant colony algorithm; clustering algorithm; malicious software identification; computer security; optimization algorithm

I. INTRODUCTION

In today’s digital age, malicious software poses a huge threat to computer systems and networks. Malware refers to software programs that implant, propagate, or execute malicious behavior. Its purpose may involve stealing personal information, disrupting system functionality, malicious dissemination, etc. [1-2]. The continuous evolution and increase of malicious software make it very difficult to protect computer systems and networks from attacks. Malware detection and identification technology face significant challenges [3-4]. Traditional signature and rule-based methods are no longer adequate to cope with the increasing number of malicious software. Due to the rapid mutation and diversity of malicious software, feature extraction and classification become extremely difficult. In addition, due to the concealment and diversity of malicious software, its detection and identification require a large amount of computing resources and time [5-6]. This study utilizes Ant Colony Optimization (ACO) to improve traditional clustering algorithms and optimize their parameters. ACO is a swarm intelligence algorithm that simulates ant behavior. It simulates the behavior of ants when searching for food and establishing pathways. ACO has been widely applied in optimization problems and has achieved significant success in solving fields such as travel salesman problems and network routing optimization [7-8]. The main contribution of the research is the proposal of an improved clustering algorithm based on ant colony optimization algorithm (ACO-CA), which is specifically designed to address the complexity of malware identification. This is the first time the ACO algorithm has been applied to the clustering problem of malicious software. Necessary adjustments and optimizations are made to the algorithm to adapt to this specific field. The ACO-CA improves the dynamics and adaptability of the clustering process by introducing ant tracking and pheromone mechanisms. This enables it to effectively handle the high-dimensional nature of malware features and the uneven distribution of samples. The algorithm proposed in this paper enhances the accuracy of malware detection and improves processing speed, demonstrating its potential for identifying malware. The deployment of the ACO-CA algorithm in practical environments is also discussed in detail, providing useful guidance for future research and application.

This study is divided into six sections. Section II provides an overview of the characteristics and threats of malware, as well as the current research status of malware identification technology. Section III proposes the hierarchical ACO-CA, which provides a detailed description of how to apply the ACO algorithm to the hierarchical aggregation clustering process. Section IV conducts empirical analysis on the performance of improved clustering algorithms and malware identification, and Section V includes discussion and analysis of experimental results. Section VI summarizes the main findings and contributions of the entire study, and explores future research directions.

II. RELATED WORKS

The research on malware identification is an important direction in the field of computer security. With the continuous increase in the number of malicious software and the continuous development of technology, the identification of malicious software has become increasingly difficult. Therefore, researchers have been striving to improve existing malware identification techniques. Hu et al. proposed a deep sub domain adaptive network with attention mechanism. The experiment showed that the average accuracy of this method reached 97.15%, and it could quickly converge without using a large number of target domain training datasets [9]. Mario et al. classified malicious software applications from system call traces. The method demonstrated high robustness in identifying infected applications and in code conversion and major avoidance techniques [10]. Yuan et al. proposed an anomaly detection method based on a dual head neural network. After filtering, the identified samples, the accuracy of malware detection increased by 8.62% and 13.12% respectively [11]. Sanjeev et al. proposed a novel malware detection architecture that utilizes image analysis and machine learning. Numerous experiments have shown that the methods of stacking global features and stacking local features have achieved testing accuracy of 98.34% and 98.23%, respectively. On the latest malware dataset in the real world, its testing accuracy was 92.75%, with a low false alarm rate [12]. Somayyeh et al. proposed a malware detection method based on short-term and short-term memory. Case studies have shown that the model can even detect new malware with an accuracy of over 90%. In addition, the model could detect malicious software by capturing 50 connection flows, with an AUC exceeding 99.9% [13]. Mahesh et al. proposed an adaptive red fox optimization method based on convolutional neural networks to detect whether malicious software applications are benign or malicious. Comprehensive experiments have shown that the detection accuracy of this method is 97.29% [14]. Gao et al. developed a practical system called HincTI for modeling network threat intelligence and identifying threat types. Compared with the most advanced baseline methods currently available, the proposed method could significantly improve the performance of threat type identification [15].

The basic version of ACO has matured and has been widely applied. With the deepening of research, many scholars have improved the efficiency and robustness of ACO by improving algorithm parameter settings, introducing heuristic information, and improving pheromone update strategies. Chen proposed a method for balancing enterprise resource information scheduling based on improved ACO. This method had good resource information balance scheduling ability and could effectively improve resource utilization [16]. Tan et al. established a mathematical model for spot welding path planning. The simulation analysis revealed that the ACO path was improved under six different parameters, resulting in an average path length of 10357.7509 millimeters. This is in contrast to the 10830.8394 millimeters obtained by traditional algorithms. The convergence analysis of improved ACO showed that its average number of iterations is 17. Therefore, the improved ACO had higher solution quality and faster convergence speed [17]. Wang et al. proposed a progressive randomization approach that combines exploration and utilization with improved ACO for automatic detection of snow melting on the surface of the Antarctic ice sheet. Further validation of six automatic weather stations indicated that the proposed method had higher accuracy [18]. Wang et al. proposed an improved ant colony resource scheduling algorithm. When the number of tasks reached 200, the proposed algorithm used 17.52% less execution time and 9.58% less resources than traditional algorithms, achieving a resource allocation rate of 91.65% [19]. Saemi et al. designed a Meta heuristic algorithm based on ACO. Compared with the sequential method, this algorithm provided a solution for comprehensive problems that reduced costs by 21.64% in a reasonable amount of time. In addition, for small problems, the average difference between the solution provided by the ACO algorithm and the optimal solution (exact method) was 2.96% [20].

In summary, research on malware identification and ACO is constantly developing and advancing, providing powerful solutions for computer security and optimization issues. The use of ACO-CA has potential in malware identification, but there is a lack of relevant content in existing research, so further research is still needed. This study aims to improve ACO and achieve better results in the field of malware detection and defense.

III. A HIERARCHICAL CLUSTERING MODEL FOR MALWARE BASED ON IMPROVED ANT COLONY

The initial section extracts the characteristics of malicious software and classifies its dynamic features through static analysis. Then, the extracted malware features are dimensionally reduced and centered through a vector feature matrix. In the second section, a clustering objective function is designed for ACO, and finally, pheromone update and probability transfer mechanisms are introduced to optimize ACO.

A. *Malware Feature Extraction and Processing Methods*

Malicious software feature extraction and processing refers to the analysis and processing of malicious software samples, extracting feature information from them, and performing corresponding processing and encoding. The purpose is to be used in security applications such as classification, detection, and protection. The common features of malware are shown in Fig. 1.

In Fig. 1, static features include file attributes (file name, path, size), file hash value, and compilation timestamp, etc. The static analysis process is shown in Fig. 2.

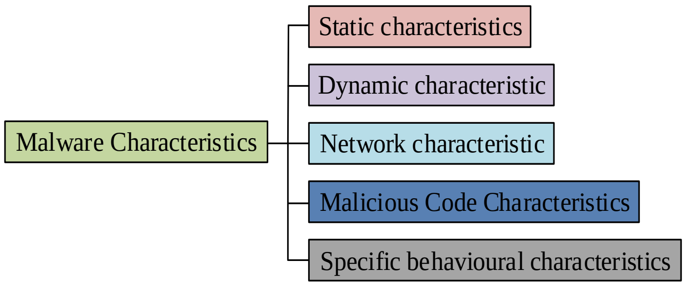

Fig. 1. Classification of common malware features.

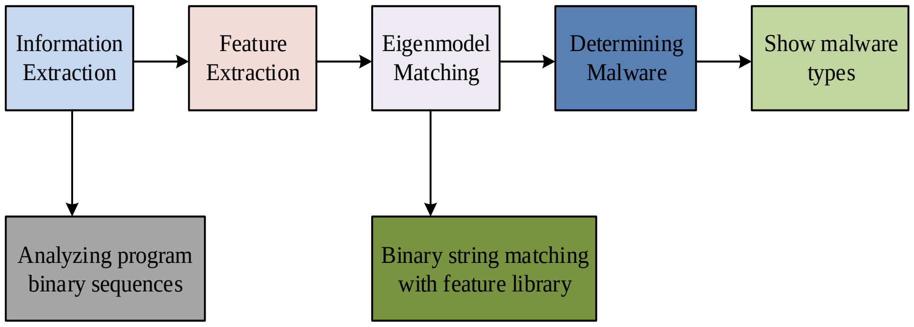

Fig. 2. Signature-based static analysis process.

In Fig. 2, the static analysis process analyzes the obtained function signature information to predict code behavior and potential defects. Dynamic characteristics refer to processes such as process behavior, system calls, registry operations, etc. Network characteristics involve network communication behavior, network traffic patterns, etc. Malicious code features include malicious code structure, invocation methods, encryption, and obfuscation. Specific behavioral characteristics refer to the encryption behavior of ransomware and the monitoring behavior of spyware. Processing malware features includes encoding, normalizing, dimensionality reduction, and other operations on the extracted features to facilitate subsequent machine learning algorithms for training and classification. The characteristics of malicious software can be represented by Hash encoding, as shown in Formula (1).

$$hash\_value = hash\_algorithm(data) \quad (1)$$

In Formula (1), $hash\_algorithm$ is the selected Hash algorithm. $data$ is the malware feature data that needs to calculate hash encoding. Assuming there is a malware feature vector X containing n eigenvalues, then the Min-Max normalization formula can be used to normalize the malware features, as shown in Formula (2).

$$normalized\_value = (value - Min(X)) / (Max(X) - Min(X)) \quad (2)$$

In Formula (2), $Min(X)$ and $Max(X)$ are the minimum and maximum values of each feature, respectively. $value$ is an eigenvalue in the feature vector X. By applying this formula, each eigenvalue can be mapped to the range of [0, 1]. Principal Component Analysis (PCA) is used to convert high-dimensional feature vectors of malicious software into low-dimensional representations, as shown in Fig. 3.

In Fig. 3, the horizontal axis represents the selected principal components, and the vertical axis represents the original data samples. Each point represents a malware sample. The original data is projected onto the selected eigenvectors after calculating the covariance matrix to obtain eigenvalues and eigenvectors, resulting in dimensionality reduced data. Assuming there are m malware samples, each with d features, and the feature vector matrix is $X(m \times d)$. The feature vector matrix is centralized as shown in Formula (3).

$$X' = X - mean(X) \quad (3)$$

In Formula (3), X' is the new centralization matrix. Then to calculate the covariance matrix of the centralization matrix X', as shown in Formula (4).

$$C = (1/m) * X'^{\wedge}T * X' \quad (4)$$

In Formula (4), X'^{T} is the transpose matrix of X'. * represents multiplication of matrices. $1/m$ is the normalization factor, ensuring that each element of the covariance matrix is within a reasonable range. Then to calculate the eigenvalues and eigenvectors of the covariance matrix, as shown in Formula (5).

$$C * v = \lambda * v \quad (5)$$

In Formula (5), λ is the eigenvector corresponding to the eigenvalues and v. Then, the eigenvalues λ are sorted in descending order. When sorting feature values, to use the argsort function in the NumPy library. The feature values are stored in a one-dimensional array eig_vals and the sorting index of the feature values are calculated, as shown in Formula (6).

$$sorted\_indices = np.argsort(eig\_vals) \quad (6)$$

In Formula (6), $sorted\_indices$ is the sorted feature value index, and then the feature values are sorted according to the sorting index to obtain Formula (7).

$$sorted\_eig\_vals = eig\_vals[sorted\_indices] \quad (7)$$

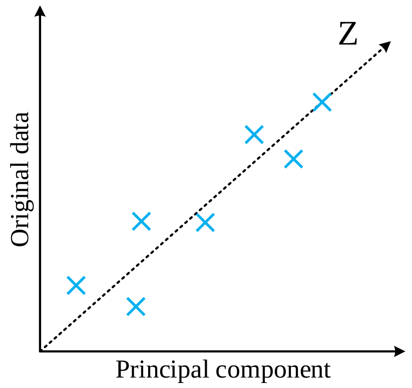

Fig. 3. Sketch of the principal component analysis of malware features.

In Formula (7), $sorted\_eig\_vals$ is the result sorted by eigenvalues. To select the eigenvector corresponding to the largest k eigenvalues, usually $k < d$, as the principal component, as shown in Formula (8).

$$V = \frac{topk\_eigvecs}{\|topk\_eigvecs\|} \tag{8}$$

In Formula (8), $topk\_eigvecs$ is the first k eigenvectors. The feature vector matrix X is multiplied by the projection matrix V to obtain the dimensionality reduced feature matrix as shown in formula (9).

$$Y = X * V \tag{9}$$

In Formula (9), the dimension of X is $(m \times d)$. The dimension of V is $(d \times k)$. The dimension of Y is $(n \times k)$. The reduced feature matrix can be used for subsequent malware feature analysis tasks, such as malware classification and anomaly detection. The study examined the effect of modifying a single parameter on the objective function while holding all other parameters constant. To systematically identify the optimal parameter combination and enhance the accuracy and efficiency of the algorithm, the grid search method was employed.

B. *Improved Ant Colony and Hierarchical Clustering Algorithm Optimization Design*

After extracting features from malicious software through feature extraction methods, ACO is used to optimize feature selection and classifier training. Finally, hierarchical clustering algorithm is used to cluster the extracted features and identify different malicious software families. The specific process is shown in Fig. 4.

Fig. 4 first extracts and processes the features of malicious software, then identifies the features of all software, and then determines whether all software has been recognized. The above steps are repeated until the ant completes the search task. Finally, to cluster the extracted features and determine the division of malware families based on the clustering results and thresholds. ACO is a heuristic optimization algorithm that simulates the foraging behavior of ant populations. Each ant selects the next location to move based on the current pheromone and heuristic information, and the ant colony routing mechanism is shown in Fig. 5.

There are two paths in Fig. 5, ABECD and ABFCD. In path BFC, as ants increase and the amount of information increases, the probability of path selection increases. In path BEC, as time increases, the amount of information decreases, and the probability of path selection decreases. To design a clustering objective function based on internal indicators, as shown in Formula (10).

$$f = \left(\frac{\sum_{i=1}^{k} D(c_i, c)}{k} \right) / \left(\frac{\sum_{j=1}^{k} \sum_{p \in C_j} D(p, c_j)^2}{k} \right) \tag{10}$$

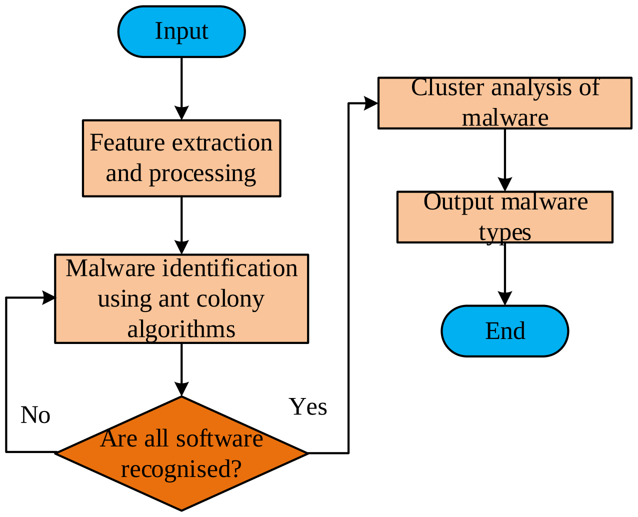
Input
Feature extraction and processing
Malware identification using ant colony algorithms
No
Yes
Are all software recognised?
Cluster analysis of malware
Output malware types
End

Fig. 4. Malware identification process.

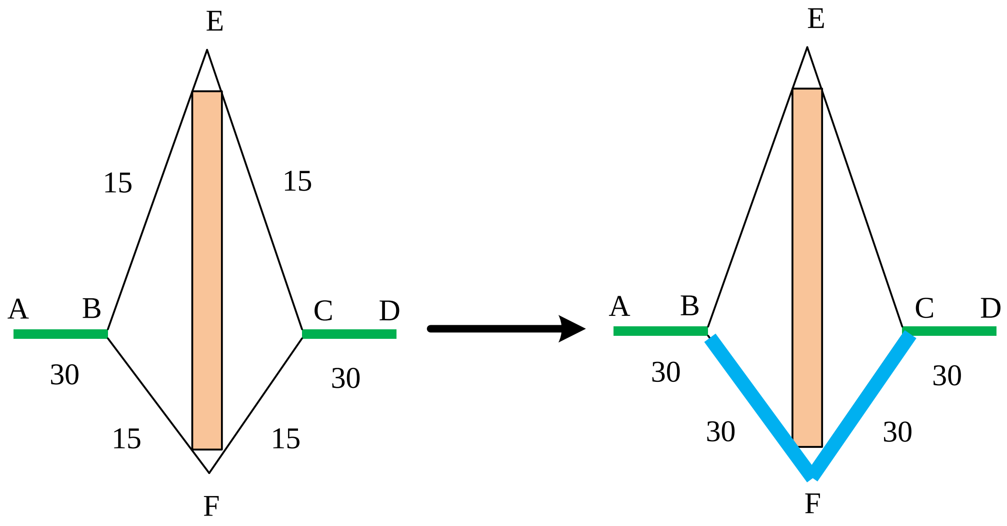

Fig. 5. Ant Colony pathfinding mechanism.

In Formula (10), c_i is the centroid of the i-th cluster. c is the centroid of all elements. C_j is the j-th cluster. D is the proximity. k is the number of clusters after clustering is completed. When clustering, not only distance information is considered, but also pheromone concentration and probability random mechanisms are introduced. The new transfer probability is Formula (11).

$$p_{ij} = \frac{u_{ij}^{\alpha} \times \left(\frac{1}{d\left(C_i, C_j\right)} \right) \beta}{\sum_{l \in Node\_Left} \left(u_{il}\right)^{\alpha} \left(\frac{1}{d\left(C_i, C_j\right)} \right) \beta} \tag{11}$$

In Formula (11), α is the pheromone concentration factor. β is the heuristic information factor. u_{ij} is the average pheromone concentration between cluster i and cluster j, as shown in Formula (12).

$$u_{ij} = \frac{\sum_{k \in C_i} \sum_{n \in C_j} \tau_{kn}}{m_i \times m_j} \tag{12}$$

In Formula (12), k and n are two points in the path. m_i and m_j are the number of ants in two clusters. τ_{kn} is the concentration of pheromones between points k and n. The algorithm process is Fig. 6.

Fig. 6 considers each point as a cluster and randomly assigns a certain number of ants to these clusters based on the proximity matrix and pheromone concentration matrix. Next, using roulette wheel to select the clusters to merge and calculate the merged result based on the objective function. If the number of iterations is not less than the set number, to output the optimal clustering merge scheme and clustering tree graph. To accelerate the convergence speed of the algorithm, the pheromone matrix is modified , as shown in Formula (13).

$$\tau_{ij}^{*} = \sigma_{ij} \tau_0 \tag{13}$$

In Formula (13), τ_{ij}^{*} is the initial pheromone concentration from element i to element j. σ_{ij} is the concentration coefficient. τ_0 is the basic pheromone concentration. After optimizing the pheromone update mechanism, it is shown in Formula (14).

$$\tau_{ij} = (1-\rho)\tau_{ij} + \varepsilon \sum_{k=1}^{m} \Delta\tau_{ij}^{k}(t) + \Delta\tau_{ij}^{*}(t) \tag{14}$$

In Formula (14), ε is the coefficient of weakness. $\Delta\tau_{ij}^{*}(t)$ is an additional pheromone on an excellent path, and the calculation Formula is (15).

$$\Delta\tau_{ij}(t) = \frac{Q}{l_{ij}} \tag{15}$$

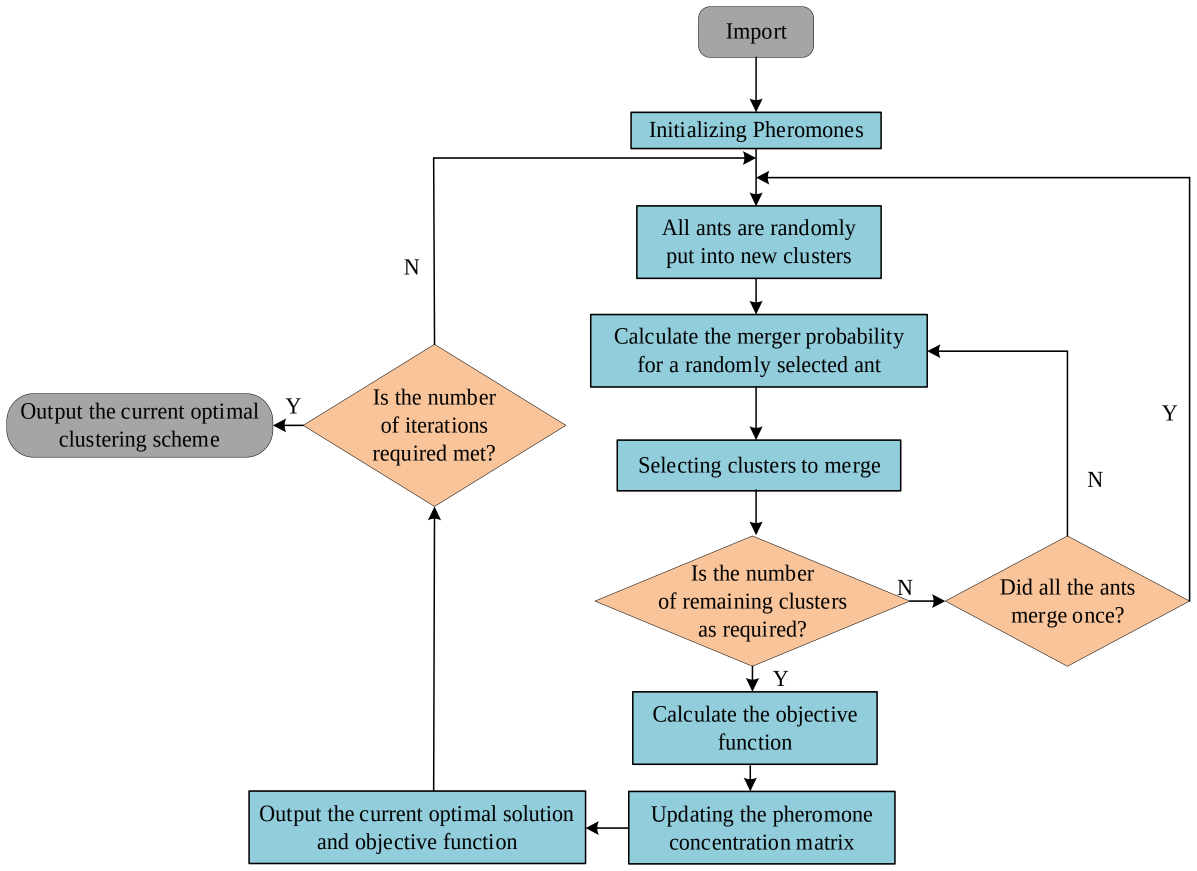
Import
Initializing Pheromones
All ants are randomly put into new clusters
Calculate the merger probability for a randomly selected ant
Selecting clusters to merge
Is the number of remaining clusters as required?
N
Did all the ants merge once?
N
Y
Y
Calculate the objective function
Updating the pheromone concentration matrix
Output the current optimal solution and objective function
Is the number of iterations required met?
N
Y
Output the current optimal clustering scheme

Fig. 6. Improvement of ant colony algorithm calculation flow.

In Formula (15), l_{ij} is the distance between two points. Q is the amount of pheromones. To simplify the parameter optimization process, the algorithm uses a probabilistic model to predict parameter performance. New parameters are then selected for testing in regions where predicted performance is improved. This approach helps balance the relationship between exploration and exploitation, leading to more efficient identification of optimal solutions.

IV. Empirical Analysis of ACO-CA Performance and Malicious Software Identification

When testing the performance of the improved ACO algorithm, the accuracy, echo value, and false alarm rate of different algorithms are first compared. Next, the accuracy and loss curves of the improved algorithm in the test and training sets are compared, and finally, the specificity of the improved algorithm and the traditional algorithm is compared. Empirical analysis compares the recognition of malicious software and CPU resource usage using different algorithms, and finally uses the proposed algorithm to monitor a certain device.

A. *Improved Algorithm Heating Performance Test Experiment*

The analysis of improving algorithm performance selected the CICAlDroid2020 dataset as the test set and CICAlDroid2019 as the training set. CICMalDroid2019 and CICMalDroid2020 are datasets used to analyze malicious Android applications, containing 5000 and 10000 Android application samples, respectively. These samples are divided into two categories: normal applications and malicious applications. Table I shows the experimental environment for this experiment.

Experiments are conducted using traditional clustering algorithms and ACO-CA on CICCalDroid2020, and their performance in indicators such as malware recognition accuracy, recall rate, and false alarm rate were compared. The results are shown in Fig. 7.

From the data in Fig. 7, ACO-CA outperforms traditional clustering algorithms in terms of accuracy, echo value, and false alarm rate. The accuracy of ACO-CA is 0.984, while the traditional clustering algorithm is 0.731. ACO-CA can more accurately classify samples and predict the categories of malware and normal software. The echo value of ACO-CA is 0.789, while the traditional clustering algorithm is 0.674. This indicates that ACO-CA performs better in terms of echo value. The false positive rate of ACO-CA is 0.345, while the traditional clustering algorithm is 0.401. The accuracy and loss curves of the improved algorithm in the test and training sets are shown in Fig. 8.

TABLE I. Experimental Hardware and Software Environment

Typology	Configurations
Operating system	Ubuntu 22.04.1
CPU Model	Intel Core i5 1240P
Random access memory (RAM)	16G
Hard disk	512G
Python	3.9.13
Python Toolkit	Numpy+Pandas+Matplotlib+Scikit-learn etc
Programming Environment	Vscode+jupyter

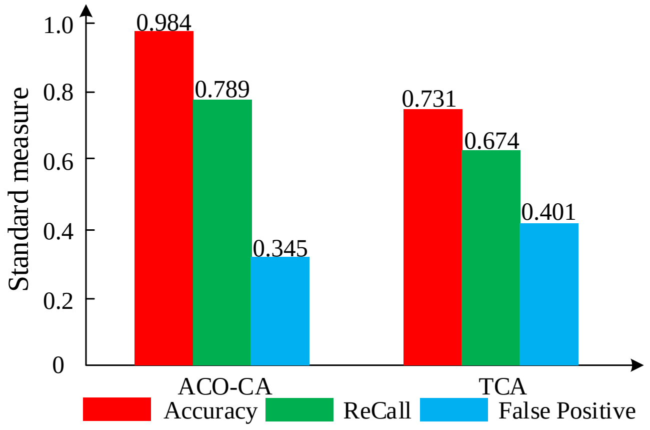

Fig. 7. Metrics performance of different clustering algorithms.

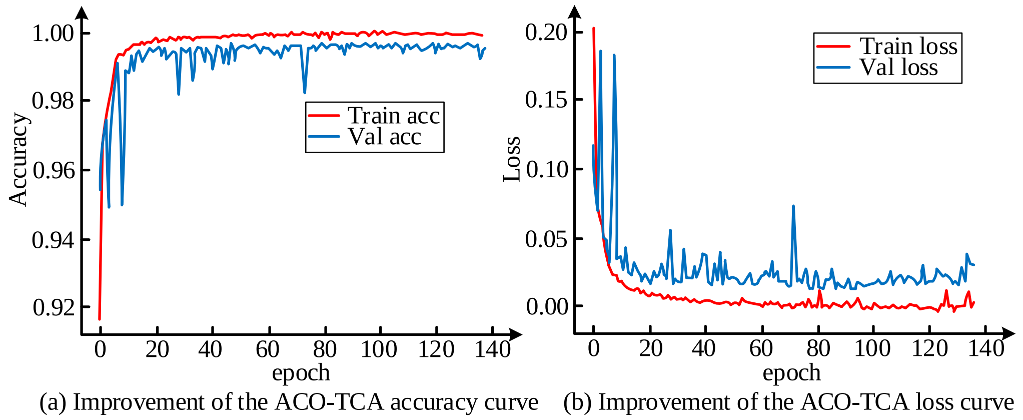

Fig. 8. Improvement of ACO-TCA accuracy versus loss curves.

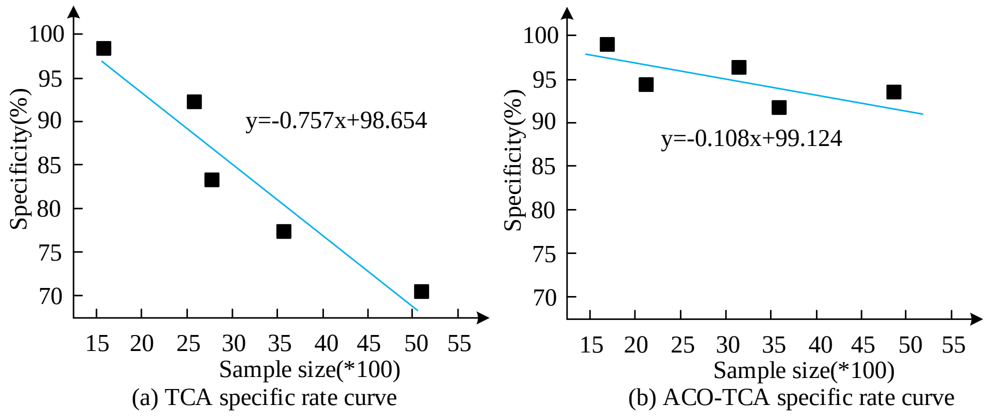

Fig. 9. Different algorithms specific rate fitting curves.

From Fig. 8 (a), the accuracy curves of the improved algorithm show an upward trend on both the training and validation sets. This indicates that as the training progresses, the algorithm gradually improves its accuracy during the learning process. Especially when the epoch is around 80, the accuracy tends to stabilize and remains at a high level. This indicates that the improved algorithm can achieve good classification results on both the training and validation sets, and has high accuracy. Secondly, from Fig. 8 (b), the loss curves of the improved algorithm on both the training and validation sets show a downward trend, and have already entered the range below 0.05 when the epoch is 20. Although there were some fluctuations afterwards, the overall level remained relatively low, not exceeding 0.1. This indicates that the improved algorithm can effectively reduce the loss rate during the training process, and a low loss rate indicates that the model can accurately predict the category of samples. The specificity between the improved algorithm and the traditional algorithm is shown in Fig. 9.

In Fig. 9 (a), the specificity decrease rate of traditional clustering algorithms is 0.757, while the specificity decrease rate of ACO-CA is 0.108. ACO-CA performs better at the rate of decrease in specificity and decreases more slowly. This indicates that ACO-CA can maintain a higher level of specificity when processing more samples. In Fig. 9 (b), the longitudinal intercept of traditional clustering algorithms is 98.654, while the longitudinal intercept of ACO-CA is 99.124. Therefore, ACO-CA has higher specificity values when the sample size is small.

B. *Empirical Experiment on Malicious Software Identification and Classification*

Malware can be classified into various types, including viruses, worms, trojans, spyware, and adware, etc. This malicious software may steal users' personal information, damage system files, and indiscriminately send spam, posing serious risks and losses to computer systems and users. The goal of malware identification and classification is to accurately identify unknown software samples as malware or legitimate software by constructing an accurate classification model. The identification of malicious software in a dataset using different algorithms is Fig. 10.

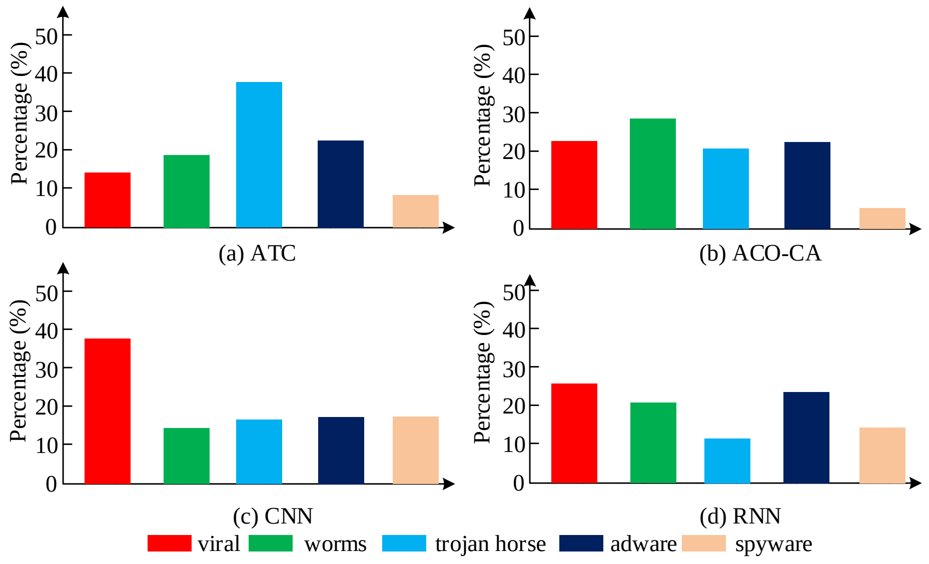

Fig. 10. Different algorithms for malware recognition statistics.

In Fig. 10 (a), it can be concluded that among traditional clustering algorithms, Trojan software has the highest recognition rate, accounting for 38% of the total, followed by advertising software and worm software. The recognition rates of viruses and spyware are relatively low, accounting for 13% and 8% respectively. In Fig. 10 (b), by using ACO-CA, the recognition rates of viruses and worm software have been improved, accounting for 22% and 28% respectively. The recognition rate of Trojan software has decreased, accounting for only 21%. The recognition rate of advertising software and spyware remains unchanged. This indicates that ACO-CA can improve the recognition ability of viruses and worm software in certain aspects. In Fig. 10 (c), Compared with traditional clustering algorithms, the CNN algorithm has a recognition rate of 39% for viruses and 16% for advertising software, which is significantly higher than other algorithms. In Fig. 10 (d), in the identification of malicious software based on RNN algorithm, the recognition rates of viruses and advertising software are relatively high, accounting for 27% and 26% respectively. The CPU resources occupied by different algorithms for recognition are shown in Fig. 11.

According to Fig. 11, based on ACO-CA and traditional clustering algorithms, the CPU resource utilization remains at a relatively low level (0.20-0.22 and 0.22-0.20) during the first 12 minutes of runtime. This may be the stage of algorithm initialization and data preprocessing, requiring less computational resources. After a running time of 12 minutes, the CPU resource utilization based on ACO-CA and traditional clustering algorithms rapidly increases, reaching levels of 0.78 and 0.75, respectively. The CPU resource utilization rate based on CNN algorithm and RNN algorithm is relatively high in the first 12 minutes of runtime (0.24 and 0.18). This may be because these two algorithms typically require more computing resources for convolution and loop operations. In the following period, the CPU resource utilization rate based on CNN algorithm and RNN algorithm gradually increases and stabilizes at a higher level (0.80). The real-time monitoring of the software operation of a certain device by ACO-CA is shown in Fig. 12.

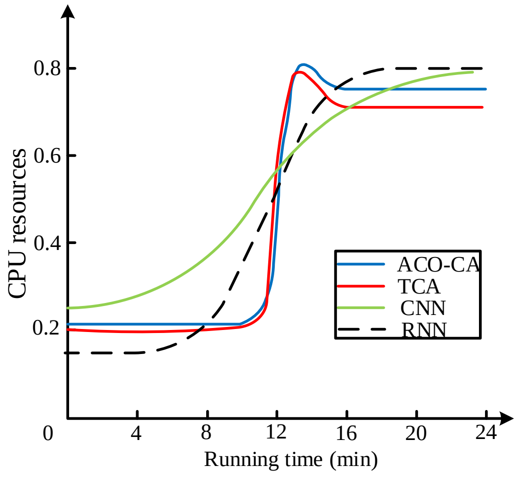

Fig. 11. Variation curve of CPU resources occupied by different algorithms.

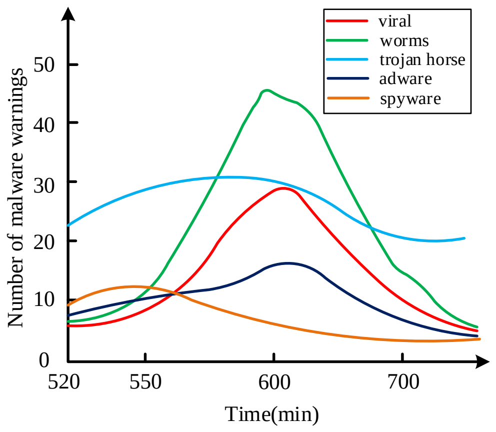

Fig. 12. Detection diagram of a device compromised by malware.

According to the data in Fig. 12, it is concluded that worm like malware has the highest number of attacks on this device. When the running time reaches 600 minutes, the number of attacks reaches 46. This may be because the security measures of the device are weak, allowing worm like malware to easily invade and attack. Trojan viruses are the second most common type of malware, with attacks fluctuating between 20 and 30 times. This may mean that the device has some protection measures in terms of security, but it is still vulnerable to Trojan virus attacks. Viruses are also a type of malware that attacks more frequently when running for 600 minutes, reaching 27 times. Advertising software and spyware have relatively fewer attacks during this period, with an average of less than 10 attacks. This may be because the purpose of advertising software and spyware is different. The former mainly obtains revenue through pop-up ads and other methods, while the latter is mainly used to monitor user activities and steal confidential information. The device may have taken certain measures to suppress the intrusion of adware and spyware.

V. Results and Discussion

Malicious software identification plays a crucial role in the field of network security, and traditional clustering algorithms have certain limitations in dealing with complex and ever-changing malicious software. Therefore, the study proposed to use the ACO algorithm to improve clustering methods, aiming to improve the accuracy and efficiency of malware detection. The algorithm has been improved to optimize the clustering process by simulating the behavioral characteristics of ant colonies. The results showed that the improved algorithm achieved a clustering accuracy of 0.984, far exceeding the traditional algorithm's 0.731. The false alarm rate has also been reduced from 0.401 in traditional algorithms to 0.345, indicating a significant improvement in reducing false positives. The accuracy improvement stabilized gradually during the training process, and the loss rate dropped below 0.05 by epoch 20. The specificity of the improved ACO algorithm decreased to only 0.108, compared to the traditional algorithm's 0.757. When processing small sample data, its specificity also showed a high level. In contrast, although CNN algorithm performed well in identifying specific categories of malware and RNN algorithm also demonstrated good classification ability, improved ACO algorithm was more outstanding in overall performance. From a resource consumption perspective, the improved ACO algorithm maintained a low CPU usage rate during the initial 12 minutes. Although it increased thereafter, the trend of higher CPU usage in the early stages and stable increase in the later stages is relatively mild as compared to the CNN and RNN algorithms. However, there are still shortcomings in the research. Although the improved ACO algorithm has shown advantages in multiple indicators, its ability to recognize different types of malware needs further improvement. Future work will focus on optimizing algorithms for universality and efficiency on large-scale datasets, providing stronger technical support for the dynamic identification of malicious software.

VI. Conclusion and Future Work

The constant increase of malware poses a serious threat to computer systems and network security. To solve this problem, the improved ACO algorithm was used to enhance the traditional clustering algorithm, aiming at improving the performance in the task of malware recognition. The study was processed in two stages. Firstly, the malware feature was extracted by static analysis, and the dynamic feature set of the malware was collected. Then the obtained features were processed by dimensionality reduction and centralized by vector eigenmatrix. Secondly, the clustering objective function suitable for ACO algorithm was designed, introducing pheromone updating and probability transfer mechanism to optimize the clustering effect. The experimental results showed that compared with the traditional clustering algorithm, the improved ACO-CA has achieved better performance in three aspects of accuracy, echo value and false positive rate. The improvement rates were 0.253, 0.115 and 0.056, respectively. The improved algorithm achieved recognition rates of 22% and 28% for virus and worm software, respectively. Compared with the traditional algorithm, the specificity decline rate was only 0.108, indicating a slow decline trend in maintaining specificity. From this, this paper had potential application value in improving clustering performance and reducing false alarms, providing a feasible technical approach for real-time monitoring of device software operation status. The proposed algorithm can be deployed in network security intrusion detection systems to monitor and identify malicious software behavior in real-time. The real-time monitoring capability of algorithms can be integrated into existing firewalls, intrusion detection systems, and intrusion defense systems. However, there are still shortcomings. Further research directions can combine ACO algorithm with other machine learning algorithms to further enhance the accuracy and generalization ability of malware identification, providing more specific deployments for practical applications.

References

[1] Nani L.Y.F, Aziah A, Masnida H. A Dynamic Malware Detection in Cloud Platform. International Journal of Difference Equations, 2020, 15(2): 243-258.

[2] Ahmad A. Packing resistant solution to group malware binaries. International Journal of Security and Networks, 2020, 15(3): 123-132.

[3] Amanul I, Fazidah O, Nazmus S, Hafiz M.H.B. Prevention of Shoulder-Surfing Attack Using Shifting Condition with the Digraph Substitution Rules. Journal of Computational and Cognitive Engineering, 2023, 1(1): 58-68.

[4] Aslan, O, Samet R. A Comprehensive Review on Malware Detection Approaches. IEEE Access, 2020, 8(1): 6249-6271.

[5] Saleh M.S, Ahmed H.E.F, Mohamed S.T, Tamer H.F, Nesrin A.A. Android Malware Prevention on Permission Based. International Journal of Applied Engineering Research, 2020, 15(1): 5-11.

[6] Mouhamed B.B, Yanjun Q, Clement K.K, Kevin M.N. Evaluation of Factors Affecting Road Maintenance in Kenyan Counties Using the Ordinal Priority Approach. Journal of Computational and Cognitive Engineering, 2023, 2(3): 260-265.

[7] Monojit D, Arnab D, Avishek B, Ujjwal K.K, Samiran C. Construction of Efficient Wireless Sensor Networks for Energy Minimization Using a Modified ACO Algorithm. International Journal of Sensors, Wireless Communication and Control, 2021, 11(9): 928-950.

[8] Kumar D, Jha V.K. An improved query optimization process in big data using ACO-GA algorithm and HDFS map reduce technique. Distributed and parallel databases, 2021, 39(1): 79-96.

[9] Hu X, Zhu C, Cheng G, Li R, Wu H, Gong J. A Deep Subdomain Adaptation Network with Attention Mechanism for Malware Variant Traffic Identification at an IoT Edge Gateway. IEEE internet of things journal, 2023, 10(5): 3814-3826.

[10] Mario L.B, Marta C, Fabrizio M.M. Data-aware process discovery for malware detection: an empirical study. Machine learning, 2023, 112(4): 1171-1199.

[11] Yuan C, Cai J, Tian D, Ma R, Jia X, Liu W. Towards time evolved malware identification using two-head neural network. Journal of information security and applications, 2022, 65(Mar): 1-11.

[12] Kumar S, Janet B, Neelakantan S. Identification of malware families using stacking of textural features and machine learning. Expert Systems with Application, 2022, 208(Dec): 1-18.

[13] Somayyeh F, Amir Jalaly B. Android malware detection using network traffic based on sequential deep learning models. Software: Practice and experience, 2022, 52(9): 1987-2004.

[14] Mahesh P.C.S, Hemalatha S. An Efficient Android Malware Detection Using Adaptive Red Fox Optimization Based CNN. Wireless personal communications: An Internaional Journal, 2022, 126(1): 679-700.

[15] Gao Y, Li X, Peng H, Fang B, Yu PS. HinCTI: A Cyber Threat Intelligence Modeling and Identification System Based on Heterogeneous Information Network. IEEE Transactions on Knowledge and Data Engineering, 2022, 34(2): 708-722.

[16] Chen S. A Balanced Scheduling Method of Smart City Enterprise Resource Information Based on Improved Ant Colony Algorithm. Journal of Testing and Evaluation: A Multidisciplinary Forum for Applied Sciences and Engineering, 2023, 51(3): 1265-1276.

[17] Tan Y, Ouyang J, Zhang Z, Lao Y, Wen P. Path planning for spot welding robots based on improved ant colony algorithm. Robotica: International journal of information, education and research in robotics and artificial intelligence, 2023, 41(3): 926-938.

[18] Wang X, Guo Z, Zhang H, Wang C, Wang Y. Snowmelt detection on the Antarctic ice sheet surface based on XPGR with improved ant colony algorithm. International journal of remote sensing, 2023, 44(1/2): 142-156.

[19] Wang Y, Liu J, Tong Y, Yang Q, Liu Y, Mou H. Resource scheduling in mobile edge computing using improved ant colony algorithm for space information network. International journal of satellite communications and networking, 2023, 41(4): 331-356.

[20] Saemi S, Komijan A.R, Tavakkoli M.R, Fallah M. Solving an integrated mathematical model for crew pairing and rostering problems by an ant colony optimisation algorithm. European Journal of Industrial Engineering, 2022, 16(2): 215-240.