Hindawi
Security and Communication Networks
Volume 2021, Article ID 5540895, 15 pages
https://doi.org/10.1155/2021/5540895

Research Article

A Lightweight Intelligent Intrusion Detection Model for Wireless Sensor Networks

Jeng-Shyang Pan,[1] **Fang Fan**,[1,2] **Shu-Chuan Chu**,[1,3] **Hui-Qi Zhao**,[2] **and Gao-Yuan Liu**[2]

[1]*College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao, 266590 Shandong, China*
[2]*College of Intelligent Equipment, Shandong University of Science and Technology, Taian, 271019 Shandong, China*
[3]*College of Science and Engineering, Flinders University, 1284 South Road, Clovelly Park SA 5042, Australia*

Correspondence should be addressed to Shu-Chuan Chu; scchu0803@gmail.com

Received 9 February 2021; Revised 21 April 2021; Accepted 23 April 2021; Published 3 May 2021

Academic Editor: Kuo-Hui Yeh

The wide application of wireless sensor networks (WSN) brings challenges to the maintenance of their security, integrity, and confidentiality. As an important active defense technology, intrusion detection plays an effective defense line for WSN. In view of the uniqueness of WSN, it is necessary to balance the tradeoff between reliable data transmission and limited sensor energy, as well as the conflict between the detection effect and the lack of network resources. This paper proposes a lightweight Intelligent Intrusion Detection Model for WSN. Combining k-nearest neighbor algorithm (kNN) and sine cosine algorithm (SCA) can significantly improve the classification accuracy and greatly reduce the false alarm rate, thereby intelligently detecting a variety of attacks including unknown attacks. In order to control the complexity of the model, the compact mechanism is applied to SCA (CSCA) to save the calculation time and space, and the polymorphic mutation (PM) strategy is used to compensate for the loss of optimization accuracy. The proposed PM-CSCA algorithm performs well in the benchmark functions test. In the simulation test based on NSL-KDD and UNSW-NB15 data sets, the designed intrusion detection algorithm achieved satisfactory results. In addition, the model can be deployed in an architecture based on cloud computing and fog computing to further improve the real-time, energy-saving, and efficiency of intrusion detection.

1. Introduction

Wireless sensor networks (WSN) provide the necessary underlying support for the Internet of Things and also build a landing platform for artificial intelligence (AI). Both of them have achieved deep integration and active promotion in WSN. The research and application of WSN have been involved in many fields, from the initial military reconnaissance to many aspects of social life, such as smart city, medical health, industrial production, environmental monitoring, and disaster warning [1]. WSN is a kind of wireless communication network that is composed of a large number of sensor nodes in a certain topological structure through self-organization. The sensor node monitors the target area or object and transmits the collected sensor data to the user along the network route [2]. WSN can break through the limitations of traditional monitoring methods, which not only significantly reduces the cost of detection, but also greatly simplifies the cumbersome process. With the rapid development of sensor technology, wireless communication technology, big data, computing intelligence, etc., the low-cost and easy-to-deploy WSN can satisfy our urgent desire to learn more about the surrounding environment or ourselves. This technology will greatly enhance the breadth and depth of our perception of the world [3].

The application scenarios of WSN are complex and changeable. Compared with the traditional wired network, it faces many unique problems and challenges. First of all, the

computing power and storage capacity of a single sensor node are quite limited, and the communication ability between nodes is weak. Furthermore, the sensor nodes are often scattered in a wide range or in a complex or even harsh physical environment, which makes it difficult or impossible to perform maintenance tasks such as energy supply. In addition, it is an open network with dynamic and random topology. So, it is necessary to carry out a series of targeted research to ensure the real-time, energy-saving, reliability, and other operational requirements of WSN [4]. As a data-centric network, more and more sensitive data are collected, stored, transmitted, and processed in WSN. Its security problem has become increasingly serious [5]. Due to the limitations and characteristics of WSN itself, the data is easy to be destroyed, stolen, or tampered with. How to protect network security effectively in the face of various network attacks is an important research topic. Unfortunately, passive defense only through firewalls, access control, and other means is not enough to prevent all the network attacks. Intrusion detection is a proactive security protection technology that can monitor the operating status of network systems and detect intrusions such as internal attacks, external attacks, or misoperations, so that the network system can intercept and respond as necessary [6]. Wired network intrusion detection technology has been relatively mature and can be divided into two types: misuse-based and anomaly-based. The prerequisite of misuse detection is that the knowledge of attack method has been acquired, and the intrusion mode has been defined in advance. Intrusion is detected by judging whether the collected data characteristics match the intrusion pattern database. Therefore, it only has a high detection rate for specific attack methods and is invalid for unknown attacks. In order to cope with the endless emergence of various attacks, anomaly detection method can be considered. This method assumes that cyber attacks are uncommon compared to normal behaviors. By comparing the captured network behavior with normal patterns, it can be judged whether an intrusion has occurred. Anomaly detection can deal with unpredictable attacks, but it needs to learn a lot of historical data for training [7]. In order to improve the detection efficiency, the introduction of AI is expected. Many scholars have tried to apply artificial neural network [8, 9], machine learning [10], evolutionary computing [11–13], etc. to the field of intrusion detection and have achieved constructive research results [14]. However, WSN has its own characteristics and limitations in terms of network scale, computing power, storage space, energy supply, communication bandwidth, and networking mode, which makes it impossible to directly use the traditional intrusion detection system (IDS) architecture. AI technology generally requires high computing power and consumes relatively large amounts of running time, storage resources, and energy. Therefore, it is necessary to make modifications and adjustments to the WSN intrusion detection model according to the actual application scenarios and user requirements and seek the balance between security, energy consumption, real-time, and other objectives [15, 16].

Obviously, WSN intrusion detection is a technical problem with multiple constraints. How to provide a feasible and effective solution is an important issue to be solved urgently. Many scholars have done fruitful work in this field [17]. Feature selection is an important and practical strategy for lightweight intrusion detection. Dimension reduction can improve the generalization performance and detection efficiency of intrusion detection. Literature [18] proposed a novel feature selection algorithm named DRFSA, combining an intelligent extension to the decision tree algorithm and convolution neural networks, to classify large volume of data in WSN. This model provides better intrusion detection accuracy, packet delivery ratio, and network throughput, while it reduces the network delay and false negative rate. The researchers also introduced a cryptographic mechanism to ensure the confidentiality and integrity of the data in the WSN and achieved encouraging results [19]. Literature [20] proposed a detection scheme for SQL injection attacks, which does not require access to the source code of the application, so it can be directly applied to the cloud environment. Literature [21] proposed a certificate-based aggregate signature scheme in WSN, which can resist forgery attacks. In addition, various machine learning and deep learning technologies are increasingly used to solve the WSN intrusion detection problem [22, 23].

This paper proposes a lightweight intelligent intrusion detection model for WSN. This model implements detection based on abnormal traffic data and can quickly and accurately discover attack behaviors in WSN. The k-nearest neighbors algorithm (kNN) is selected as the classifier. kNN is simple to implement and easy to understand. It supports nonlinear problems well and can provide relatively robust recognition results. The time complexity of the kNN is lower than that of the support vector machine (SVM) [24, 25]. Compared with naive Bayes algorithm [26], kNN has no hypothesis on data and is not sensitive to outliers. Therefore, compared with other machine learning algorithms, KNN meets the requirements of lightweight data classification. In order to further improve the classification effect, this paper uses evolutionary algorithm to optimize kNN. The selected evolutionary algorithm is the sine cosine algorithm (SCA). Among many metaheuristic optimization algorithms, SCA has low computational complexity, simple parameters, and good optimization performance. Taking into account the many limitations of WSN intrusion detection, the compact mechanism is applied to SCA (CSCA), which greatly reduces the time and space occupied in the optimization process. In order to ensure that the accuracy requirements are met, a polymorphic mutation strategy (PM) is designed, and an improved version of SCA is proposed (PM-CSCA). The organic combination of kNN and PM-CSCA constitutes a lightweight intelligent intrusion detection model for WSN. On the one hand, the intelligent detection is realized by means of evolutionary computation and machine learning; on the other hand, the computational burden of evolutionary algorithm is greatly reduced, so as to ensure the lightweight of the designed intrusion detection model.

This article is organized as follows: the second part is related work, introducing the SCA and kNN used in the intrusion detection algorithm proposed in this paper. The third part introduces the architecture of the intrusion detection system. The fourth part is the design of intrusion detection algorithm, including the improvement of SCA,

and how to combine it with kNN. The fifth part is the simulation results and discussion. The last part is the conclusion and future work.

2. Related Works

2.1. Sine Cosine Algorithm (SCA). SCA is a metaheuristic swarm intelligence optimization algorithm. The algorithm has a concise structure, has fewer parameters, and is easy to understand and implement. The search trajectory for the optimal solution is mainly affected by the sine and cosine functions [27–29].

The algorithm first initializes the population X, that is, to create N random candidate solutions $X_i (i = 1, 2, \ldots, N)$. They are then guided to move through the search space using mathematical models based on sine and cosine functions. The optimization process is divided into two stages: global exploration and local exploitation. The formula for updating the position of the solution is as follows:

$$\begin{cases} X_i^{t+1} = X_i^t r_1 * \sin(r_2) * \left| r_3 P_i^t - X_i^t \right|, & r_4 \geq 0.5, \\ X_i^{t+1} = X_i^t r_1 * \cos(r_2) * \left| r_3 P_i^t - X_i^t \right|, & r_4 < 0.5, \end{cases} \tag{1}$$

where t is the current number of iterations, P_i^t is the position of the current optimal solution in the i – th dimension, and $|\cdot|$ represents the absolute value. There are only four parameters involved here: r_1, r_2, r_3 and r_4. $r_2 \in [0, 2\pi]$, which controls the distance the solution moves each time. $r_3 \in [0, 1]$, which gives a random weight to the current optimal solution. $r_4 \in [0, 1]$, which controls the switching between the sine and cosine update modes to ensure the same probability of using both. The above three parameters are random numbers that obey a normal distribution within their respective ranges. The parameter r_1 determines the direction of movement. When $r_1 < 1$, the solution will move to the area between the current position and the target position to exploit the local potential space. When $r_1 > 1$, the solution is to move away from the current optimal position to explore a larger search space. r_1 decreases linearly as the number of iterations increases, realizing the transition from exploration to exploitation. The updated formula of r_1 is shown in equation (2). Generally, $a = 2$, and T represents the maximum number of iterations.

$$r_1 = a - t\frac{a}{T}. \tag{2}$$

2.2. The k-Nearest Neighbors Algorithm (kNN). kNN algorithm is commonly used in data mining and machine learning. As one of the simplest classification algorithms, kNN is widely used in many fields. The core idea is that, in the feature space, if most of the k samples closest to a sample belong to a certain category, then this sample also belongs to this category and has all its characteristics. So, only the category of the k most similar samples is used to determine the category of the pending sample when making a classification decision [30, 31]. The implementation method is that all samples are mapped to points in D-dimensional space; k known samples nearest to the unknown sample are selected as reference, and the distances between them are calculated, respectively; according to the majority voting rule, the unknown sample is classified into the category of most of its k-nearest neighbors. Obviously, kNN algorithm mainly considers three elements: the value of K, the way of distance measurement, and classification decision rules. The majority voting method is usually used to make decisions. The focus is usually on the choice of k value and the measurement of distance.

As the only parameter, the value of k has a crucial impact on the prediction results of kNN [32]. If k is relatively small, the approximate error of learning will decrease, but the estimation error will increase, and it is easy to learn noise. In severe cases, the model becomes complicated, and overfitting occurs. Similarly, if the k is large, the model will become too simple and underfit, which will also lead to inaccurate predictions. In actual engineering practice, k is generally selected by cross-validation. There is no fixed experience to guide the setting of k [33]. This has caused inconvenience in using the kNN algorithm.

We also need to pay attention to the distance measurement in the sample space. The shorter the distance, the higher the similarity between the two sample points, and conversely, the lower the similarity. The commonly used distance measurement methods are Minkowski Distance, Euclidean Distance, Manhattan Distance, Chebyshev Distance, Mahalanobis Distance, etc.

Suppose that there are two samples x_i and x_j in the D-dimensional feature space, which are expressed as $x_i = (x_{i1}, x_{i2}, \ldots, x_{i\,D})$ and $x_j = (x_{j1}, x_{j2}, \ldots, x_{j\,D})$. The distance between the two samples is denoted as $d(x_i, x_j)$. kNN classifiers generally use Euclidean distance to measure the similarity between samples, as shown in

$$d\left(x_i, x_j\right) = \sqrt{\sum_{k=1}^{D} \left(x_{ik} - x_{jk}\right)^2}. \tag{3}$$

But in the process of classification, the importance of features is often different. Some features are strongly correlated with the classification results, some are weakly correlated, and some are even negatively correlated. If the distance between samples is largely dominated by weakly correlated or irrelevant features, it will easily lead to confusion in classification. To solve this problem, a certain weight $w_k (k = 1, 2, \ldots, D)$ can be assigned to each feature dimension to express its importance. So, the distance between samples can be transformed into the following formula:

$$d\left(x_i, x_j\right) = \sqrt{\sum_{k=1}^{D} w_k \left(x_{ik} - x_{jk}\right)^2}. \tag{4}$$

As a popular machine learning algorithm, kNN has been successfully applied in many fields [34, 35]. Some literatures try to improve it, mostly around the adjustment of parameter k [36, 37]. In fact, there is no universal experience in the determination of k, the selection of distance function, or

the setting of distance weight. All of these should be based on the distribution of samples, the characteristics of data, and the needs of analysis. This can be regarded as a typical optimization problem. With the help of the optimization ability of metaheuristic algorithm, a more reasonable and effective kNN classification model can be constructed [38].

3. WSN Intrusion Detection System Architecture

Intrusion detection is a security mechanism that collects information from several key nodes in the network system and analyzes it to try to find out whether there is any behavior that violates the security policy or signs of being attacked. The data in WSN shows an explosive growth trend. This requires high data processing capabilities, and intrusion detection also requires sufficient computing power.

The cloud computing platform has powerful computing and storage capabilities, as well as open, flexible, and shared characteristics, which provides a new research idea for WSN to break through the bottleneck restricting its development. In order to reduce the burden of importing and exporting data from the cloud and relieve the pressure of bandwidth shortage, fog computing can be further introduced. As a new generation of distributed computing, fog computing is closer to the edge of the network, providing space for a wider range of nodes to access. Comprehensive utilization of cloud computing and fog computing can achieve efficient collaborative computing. The powerful data processing and storage capabilities of the cloud computing platform provide technical support for big data analysis of WSN.

The intrusion detection system designed in this paper is deployed in the network architecture that combines cloud computing and fog computing, which can give full play to its advantages and better meet the data security requirements of WSN. The intrusion detection model can be deployed on the cloud server. Fog computing can be implemented by sink nodes with rich resources, which can independently assist the cloud to complete data processing, storage, and other tasks. WSN generally adopts hierarchical network structure and is divided into several clusters. The common sensor nodes in the cluster collect data and send it to the cluster heads, which transmit the data to the fog computing virtual network composed of sink nodes in a multihop manner. Figure 1 shows the architecture of the above WSN intrusion detection system.

4. Proposed Works

4.1. The Improvement of SCA. SCA is less computationally expensive compared with many other optimization algorithms. It is a reasonable choice for solving optimization problems that require low computational complexity and high real-time performance. In order to further improve the convergence speed of SCA, this paper uses the compact mechanism to make the algorithm more lightweight. Compact SCA (CSCA) can greatly reduce the computing load, but it will inevitably lose optimization accuracy to a certain extent. To solve this problem, a polymorphic mutation strategy (PM) is proposed to enrich the diversity of population and compensate for the loss of precision. The framework structure of PM-CSCA is shown in Figure 2. In this part, the main ideas and implementation schemes of the proposed PM-CSCA are described in detail.

4.1.1. Compact SCA (CSCA). Compact is an optimization mechanism of swarm intelligence algorithm. After compact processing, the memory requirement of the algorithm will be significantly reduced [39, 40]. Because this technology will greatly alleviate the computational burden of the population-based metaheuristic algorithm, it is particularly suitable for devices with limited computing power and scarce storage space, such as sensor nodes, wearable devices, and embedded devices. SCA is an intelligent optimization algorithm based on population. The optimization process is as follows: N solutions are randomly generated in the D-dimensional space, and the positions of the solutions are constantly updated in the iterative process to realize the evolution of the population and finally find the global optimal solution. When the number of solutions is large, or the dimensionality is high, this calculation mode consumes more computing power. In application scenarios with high real-time requirements or limited storage space, the optimization algorithm needs to make necessary adjustments. The main idea of compact technology is to transform the original population into the form of a probability model that reflects its distribution characteristics. All operations on the original population are also transferred to its probability model [41, 42]. Since the number of variables and storage space required by the probabilistic model are far less than the original population, the algorithm runs more efficiently in time and space. The data structure of perturbation vector (PV) is usually used to describe the macroscopic probability distribution of the population: $\mathrm{PV}^t = [\mu^t, \sigma^t]$. Here, μ and σ are the mean and standard deviation of PV, respectively, and t represents the current iteration number. Each pair of μ and σ in PV corresponds to a probability density function (PDF) [43] and is updated with the iteration of the algorithm. Generally, PDF is a truncated normal distribution in the interval [-1, 1], and the calculation formula is as follows:

$$\mathrm{PDF}_i(x) = \frac{\sqrt{2/\pi}e^{\left(-(x-\mu_i)^2/2\sigma_i^2\right)}}{\delta\left(\mathrm{erf}\left(\mu_i + 1/\sqrt{2}\sigma_i\right) - \mathrm{erf}\left(\mu_i - 1/\sqrt{2}\sigma_i\right)\right).} \quad (5)$$

It can be seen that PDF is a function of μ and σ. Among them, $x \in [-1, 1]$, erf represents error function, and i means dimension. Next, the cumulative distribution function (CDF) corresponding to the PDF can be obtained. The calculation method is as follows:

$$\begin{aligned}\mathrm{CDF} &= \int_{-1}^{x} \mathrm{PDF}\mathrm{d}x \\ &= \int_{-1}^{x} \frac{\sqrt{2/\pi}e^{\left(-(x-\mu)^2/2\sigma^2\right)}}{\sigma\left(\mathrm{erf}\left(\mu + 1/\sqrt{2}\sigma\right) - \mathrm{erf}\left(\mu - 1/\sqrt{2}\sigma\right)\right)}\mathrm{d}x.\end{aligned} \quad (6)$$

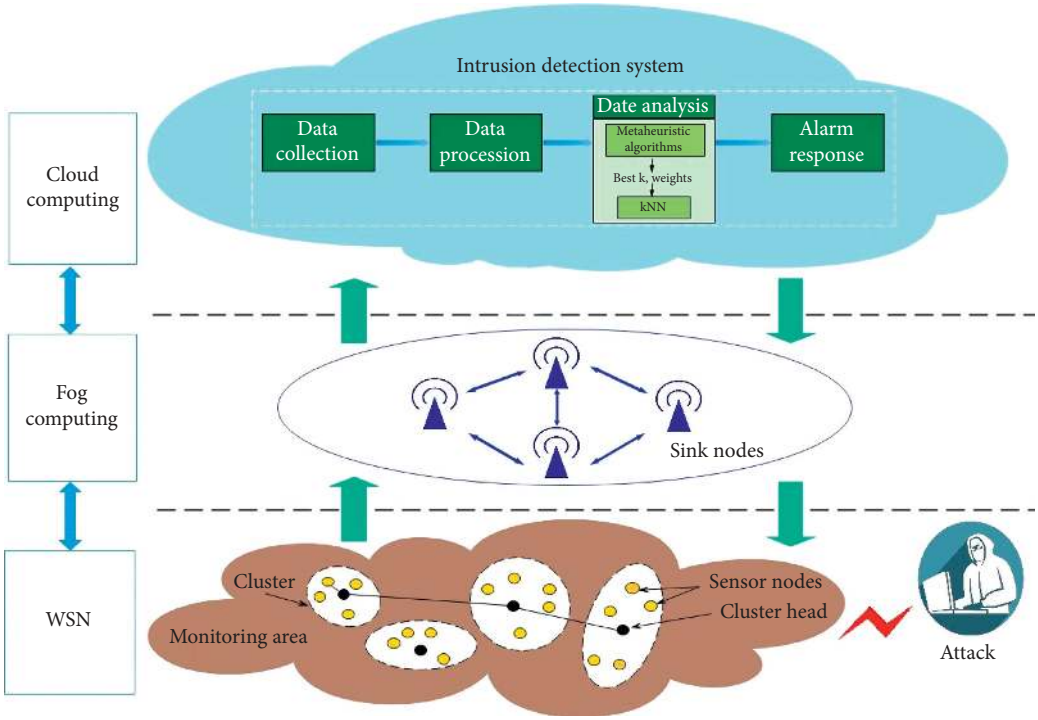


Figure 1: WSN intrusion detection system.

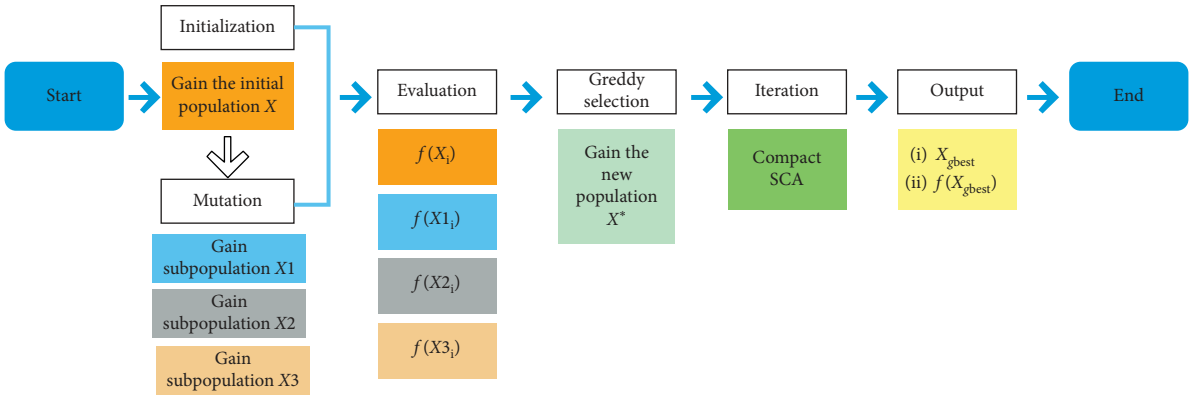


Figure 2: Framework of PM-CSCA.

Since PDF is a truncated normal distribution in the interval [−1, 1], the CDF range is from 0 to 1. With the inverse function of CDF, a virtual solution y can be obtained by using PV:

$$y = \sqrt{2}\sigma \mathrm{erf}^{-1}\left(-\mathrm{erf}\left(\frac{\mu+1}{\sqrt{2}\,\sigma}\right) - x\mathrm{erf}\left(\frac{\mu-1}{\sqrt{2}\,\sigma}\right) + x\mathrm{erf}\left(\frac{\mu+1}{\sqrt{2}\,\sigma}\right)\right) + \mu, \tag{7}$$

where $y \in [-1, 1]$, erf^{-1} is the inverse function of erf, and x is a random number between $[0, 1]$. It is necessary to map the virtual solution y to the solution y_{ds} of the decision space. Assuming that, in the D-dimensional decision space, the upper and lower limits of a certain dimension are ub and lb, respectively. y can be mapped to y_{ds} using

$$y_{ds} = y \times \frac{1}{2}(ub - lb) + \frac{1}{2}(ub + lb), \tag{8}$$

y_{ds} then attempts to move using equation (1). Evaluate the quality of the position before and after the movement, and record them as winner and loser, which are used to update the PV. Please see equations (9) and (10) for details.

$$\mu_i^{t+1} = \mu_i^t + \frac{1}{N_p}(\mathrm{winner}_i - \mathrm{loser}_i), \tag{9}$$

$$\sigma_i^{t+1} = \sqrt{\left(\sigma_i^t\right)^2 + \left(\mu_i^t\right)^2 - \left(\mu_i^{t+1}\right)^2 + \frac{1}{N_p}\left(\mathrm{winner}_i - \mathrm{loser}_i\right)^2}. \tag{10}$$

Among them, N_p is the number of solutions in the virtual population. In the process of updating PV, the global

optimal position is updated synchronously, and then the next iteration is carried out. With the help of compact mechanism, the original population is greatly reduced in size, and considerable benefits are achieved in both time and space [44–46]. However, due to the use of approximate probability distribution to simulate the real distribution of data, it is inevitable to bring the risk of loss of optimization accuracy, resulting in the occurrence of local traps or missing the global optima.

4.1.2. Polymorphic Mutation Strategy (PM). In order to make up for the possible loss of precision in compact SCA, a polymorphic mutation strategy (PM) is proposed. Based on the SCA initial population, a variety of distribution functions are introduced to realize polymorphic variation, and then the population with better quality is obtained through greedy selection. This can effectively increase the diversity of the population and create more opportunities for covering potential search areas, thereby improving the optimization accuracy. Three distribution functions are used here: Gaussian distribution, Cauchy distribution, and Levy′ distribution. Gaussian distribution is a kind of thin-tailed distribution, which is an important probability distribution in statistics. It is often used to represent an uncertain random variable. Cauchy distribution belongs to fat-tailed distribution, and the possibility of extreme values is greater than that of Gaussian distribution. Among all the distributions, the generalized Cauchy distribution has the largest spreading characteristic. Levy′ distribution can be approximated as heavy-tailed distribution. It can be used to generate Levy′ flight, that is a random walk with relatively high probability of having a larger stride. So, the search efficiency of Levy′ flight is better in the unknown environment or in large space [47].

In PM strategy, the population X initialized by SCA is randomly divided into three subpopulations: $X1, X2, X3$. Generate three variables between [0,1]: G, C, L, which obey different probability distributions: $G \sim N(\mu, \sigma^2)$, $C \sim C(\mu, \sigma^2)$, $L \sim \text{Levy}'(\lambda)(\text{Levy}' \sim u = t^{-\lambda}, 1 < \lambda \leq 3)$. Perform mutation based on Gaussian distribution on $X1$ to obtain a new subpopulation X_G, as shown in equation (11). In the same way, mutations based on Cauchy distribution and Levy′ distribution are applied to $X2$ and $X3$, respectively; and X_C and X_L are obtained according to equations (12) and (13).

$$X_G = X1 + X1 \otimes G, \tag{11}$$

$$X_C = X2 + X2 \otimes C, \tag{12}$$

$$X_L = X3 + X3 \otimes L, \tag{13}$$

Here, $G \sim N(0,1)$, $C \sim C(1,0)$, $L \sim \text{Levy}'(\lambda)$ $(\text{Levy}' \sim u = t^{-1.5})$. The product $\otimes$ means entry-wise multiplications. According to the fitness value obtained by the evaluation function $f(\cdot)$, all solutions from the population X, $X1$, $X2$ and $X3$ are sorted, and the better population X^* is obtained by greedy selection.

The computational complexity of the proposed PM-SCA depends on the following processes: initial population, polymorphic mutation, fitness evaluation, greedy selection, update population, and compact mechanism. Suppose that the number of solutions is n, the dimension is d, and the number of iterations is t. The computational complexity of initializing n d-dimensional solutions is $O(n \times d)$. The computational complexity of evaluating all solutions is $O(t \times n)$. The complexity of greedy selection is $O(n \times \log n)$. The computational complexity of updating all solutions is $O(t \times n \times d)$. Among them, the computational complexity of polymorphic mutation is $O(1)$, and the compact mechanism hardly brings about an increase in computational complexity. In general, the computational complexity of PM-SCA is the same as that of original SCA.

The pseudocode of PM-CSCA is shown in Algorithm 1. When the maximum number of iterations max_iter is reached, or other termination conditions are met, the global optimal solution x_{gbest} and its corresponding fitness value $f_{g\text{best}}$ are output.

4.1.3. Experiment Results. In order to test the performance of the algorithm, this part uses benchmark functions to carry out comparative experiments in the five algorithms of PM-CSCA, CSCA, SCA, Particle Swarm Optimization (PSO), and Whale Optimization Algorithm (WOA). 12 typical benchmark functions are selected here, including 3 unimodal functions ($F_1 \sim F_3$), 3 multimodal functions ($F_4 \sim F_6$), and 6 complex functions ($F_7 \sim F_{12}$), as shown in Table 1.

For the purpose of measuring the performance of the algorithm in a comprehensive and objective way, the algorithm runs independently 30 times in each experiment, recording the best value, average value (Avg), and standard deviation (Std), respectively. Please refer to Table 2 for specific data, and the best results have been marked in bold. The convergence curves of the benchmark functions are shown in Figure 3.

In the test of the three types of benchmark functions, PM-CSCA has achieved an absolute advantage in the algorithms participating in the comparison. The performance is particularly prominent in the optimization of complex functions. All indicators of the 6 complex functions ($F_7 \sim F_{12}$) have got the first place. PM-CSCA shows good optimization strength and reliable stability.

4.2. Combination of PM-CSCA and kNN. kNN parameter k and distance weight w_k determine the classification effect to a large extent. However, these aspects usually depend on the subjective decision of users, which brings great uncertainty to the performance of the algorithm. The PM-CSCA proposed in this article can be used to optimize the relevant parameters of kNN to obtain the best or approximately best configuration of the classifier.

The samples in the D-dimensional feature space correspond to the N solution vectors of the evolutionary algorithm: $X_i (i = 1, \ldots, N)$, the specific form is shown in equation (14). The first dimension represents the

Initialize the parameters related to the algorithm: ub, lb, Dim, max_iter, PV (μ, σ);
Generate initial population X containing N individual X_i $(i = 0, 1, 2, 3, \ldots, N)$;
Divide X into three subpopulations $X1$, $X2$, $X3$;
Realize the mutation of three subpopulations by using equations (11)–(13), respectively;
Evaluate each individual by the objective function;
Greedy selection: select N individuals from X, $X1$, $X2$ and $X3$ using greedy strategy, and get new population X^*;
Do
  Update SCA parameter: r_1, r_2, r_3 and r_4;
  Get y_1 from PV by equations (5)–(8);
  Update the y_1 by SCA to get y_2;
  Evaluate y_1 and y_2 by the objective function to get [winner, loser];
  for $i = 1$:Dim
    Update PV via by equations (9) and (10);
    if $f_{\text{winner}} < f_{\text{gbest}}$
      Update the best solution obtained so far;
  end
while $(t < \max\_\text{iter})$ or (get the expected function value);
Return the best solution obtained so far as the global optimum;

Algorithm 1: Pseudocode of PM-CSCA.

Table 1: Benchmark functions for testing.

Function	Dimension	Range	$F_{\min}$
$F_1(x) = \sum_{i=1}^{\pi} x_i^2$	20	$[-100, +100]$	0
$F_2(x) = \max_i\{\lvert x_i\rvert, 1 \le x \le n\}$	20	$[-100, +100]$	0
$F_3(x) = \sum_{i=1}^{n} i x_i^4 + \text{random}[0, 1)$	20	$[-1.28, +1.28]$	0
$F_4(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$	20	$[5.12, +5.12]$	0
$F_5(x) = (1/4000)\sum_{i=1}^{\pi} x_i^2 - \prod_{i=1}^{n} \cos(x_i/\sqrt{i}) + 1$	20	$[-32, +32]$	0
$F_6(x) = (\sum_{i=1}^{5} i * \cos(i+1)x_1 + i) * (\sum_{i=1}^{25} i * \cos((i+1)x_2) + i)$	20	$[-5.12, +5.12]$	0
$F_7(x) = ((1/500) * \sum_{i=1}^{25} (1/i + \sum_{i=1}^{2} (x_j - x_{ij})))$	20	$[-65, 65]$	0
$F_8 = 4 * x_1^2 - 2.1 * (x_1^6/3 + x_1 * x_2) - 4 * x_2^2 + 4 * x_2^4$	2	$[-5, +5]$	0
$F_9(x) = [1 + (x_1 + x_2 + x_3)^2 * (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)]$ $\times(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)$	2	$[-2, +2]$	3
$F_{10}(x) = -\sum_{i=1}^{4} [(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	$[-10, +10]$	−10.1532
$F_{11}(x) = -\sum_{i=1}^{7} [(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	$[-10, +10]$	−10.4028
$F_{12}(x) = -\sum_{i=1}^{10} [(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	$[-10, +10]$	−10.5363

parameter K of kNN, which can be set as a random integer within a certain range as required. $w_{ij} \in [0, 1]$, the random number represents the $j-$th distance weight in the $i-$th solution. Evolutionary algorithm will continuously search and iterate under the guidance of the objective function and finally output the optimal solution or the approximate best [48–51], that is, the most suitable related parameters of kNN.

$$X_i = \left[k_i, w_{i1}, w_{i2}, \ldots, w_{ij}, \ldots, w_{i\,D}\right], \quad i = 1, \ldots, N, j = 1, \ldots, D. \tag{14}$$

5. Simulation Results and Discussion

Machine learning usually uses the following four criteria to evaluate the performance of the model: the true positive (TP), true negative (TN), false positive (FP), and the false negative (FN). In the field of intrusion detection, their specific meanings are as follows: TP is the number of actual attack records classified as attacks, TN is the number of actual normal records classified as normal, FP is the number of actual normal records classified as attacks, and FN is the number of actual attack records classified as normal. They are also used to calculate a variety of performance evaluation indicators, such as detection rate (DR), false alarm rate (FAR), and accuracy rate (ACC). The calculation methods are as shown in the equations (15)–(17).

$$\text{DR} = \frac{\text{TP}}{(\text{TP} + \text{FN})}, \tag{15}$$

$$\text{FAR} = \frac{\text{FP}}{(\text{FP} + \text{TN})}, \tag{16}$$

$$\text{ACC} = \frac{(\text{TP} + \text{TN})}{(\text{TP} + \text{FN} + \text{FP} + \text{TN})}, \tag{17}$$

DR represents the probability of positive prediction among samples with normal real value. FAR is the probability of positive prediction among samples with abnormal real values. ACC is to divide the number of samples with correct prediction by the total number of samples, indicating the

Table 2: Results of PM-CSCA, CSCA, SCA, PSO, and WOA on 12 benchmark functions.

Function	Algorithm	Best value	Avg	Std
F_1	PM-CSCA	**1.49E − 95**	**1.62E − 94**	**3.62E − 21**
	CSCA	6.02E − 19	4.07E − 03	9.37E + 02
	SCA	7.04E − 17	1.70E − 19	4.03E − 19
	PSO	201.2388	8.30E + 01	3.00E + 01
	WOA	7.91E − 20	7.41E − 19	1.03E − 17
F_2	PM-CSCA	**2.79E − 08**	**5.91E − 08**	**3.93809E − 07**
	CSCA	1.70E − 06	3.22E − 01	7.330382517
	SCA	0.00010521	4.39E − 07	3.81943E − 07
	PSO	6.0981	5.82E + 00	1.426012414
	WOA	10.1124	9.35E − 02	0.004428296
F_3	PM-CSCA	**0.0024941**	**0.000918692**	**0.000285841**
	CSCA	0.0030472	0.811201	0.429978878
	SCA	0.010223	0.000496352	0.000630681
	PSO	0.076345	0.01382486	0.002997254
	WOA	0.0032755	0.000493716	0.007144592
F_4	PM-CSCA	1.97E − 11	1.74E − 04	4.49035E − 05
	CSCA	4.58E − 09	6.23E − 02	20.2012566
	SCA	3.52E + 01	3.72E + 00	0.40291051
	PSO	42.6913	2.75E + 01	8.681105181
	WOA	**0**	**0.00E + 00**	**0.000095952**
F_5	PM-CSCA	**0**	**0.070084961**	0.234385246
	CSCA	0.55431	0.4014294	16.96738349
	SCA	0.35735	0.018086667	0.171485471
	PSO	3.0755	2.14161	0.388616247
	WOA	0.12531	0.0712398	**0.019007278**
F_6	PM-CSCA	**0.022866**	**0.0694455**	**0.015782176**
	CSCA	0.10923	0.115816364	1738716.553
	SCA	0.10679	0.0784325	0.022381946
	PSO	8.9794	4.19622	1.311501322
	WOA	0.14293	**0.001379359**	0.000136212
F_7	PM-CSCA	**0.99867**	**0.998402**	**0.59335955**
	CSCA	1.0924	1.70102	0.966604051
	SCA	2.9821	1.401698	0.792971048
	PSO	1.993	0.998402	9.2957E − 05
	WOA	2.9821	1.791716	0.907953884
F_8	PM-CSCA	**0.00076939**	**0.001097476**	**0.000323742**
	CSCA	0.0015264	0.00461873	0.003881268
	SCA	0.0015936	0.000929303	0.00040039
	PSO	0.001016	0.001423611	0.0004534
	WOA	0.0014995	0.001104929	0.000151675
F_9	PM-CSCA	**3**	**3**	**2.22045E−16**
	CSCA	3.0003	3.88066	0.007128226
	SCA	3.0001	3.0003	4.58258E − 05
	PSO	3.0033	3.00784	0.000652993
	WOA	3.0001	**3**	3.68258E − 05
F_{10}	PM-CSCA	**−3.8499**	**−3.85357**	**0.000224499**
	CSCA	−3.8544	−3.80696	0.854785298
	SCA	−3.8317	−3.83598	0.000801249
	PSO	−3.6506	−3.79914	0.006526132
	WOA	−3.8074	−3.75664	0.001567945
F_{11}	PM-CSCA	**−4.9998**	**−4.35345**	**0.000961301**
	CSCA	−2.9376	−3.68698	0.077142766
	SCA	−4.5372	−3.8552	0.002753834
	PSO	−1.9555	−3.81817	0.046047219
	WOA	−3.7214	−3.86085	0.010410014

Table 2: Continued.

Function	Algorithm	Best value	Avg	Std
F_{12}	PM-CSCA	**−4.9514**	**−4.80574**	**0.201138842**
	CSCA	−0.94657	−1.89185	0.998811381
	SCA	−4.7207	−3.995813	1.204583359
	PSO	−1.4388	−2.313657	0.934009458
	WOA	−2.4202	−2.24961	0.706751911
Statistics of the number of wins	PM-CSCA	**11**	**10**	**10**
	CSCA	0	0	0
	SCA	0	0	0
	PSO	0	0	0
	WOA	1	2	2

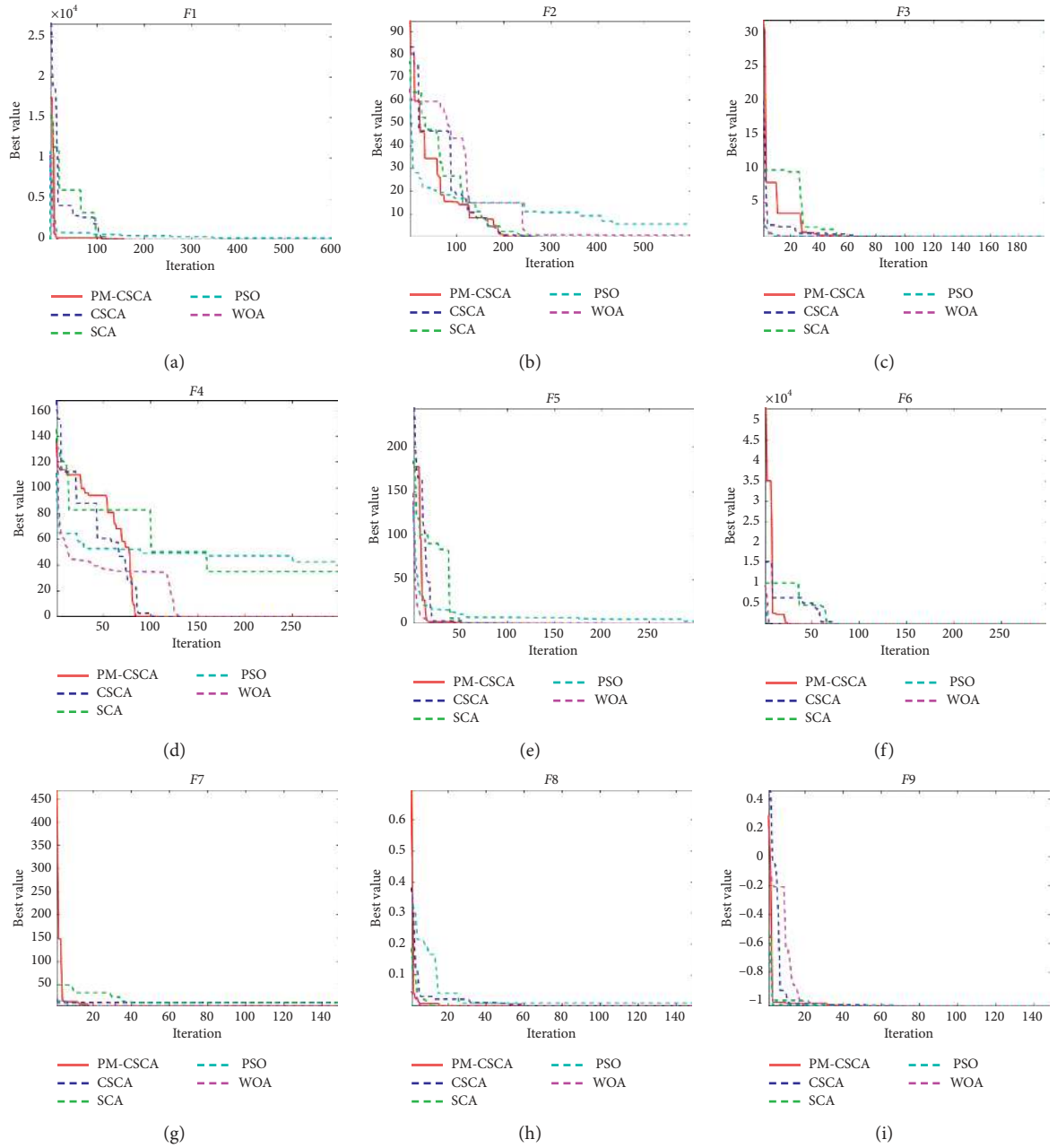


Figure 3: Continued.

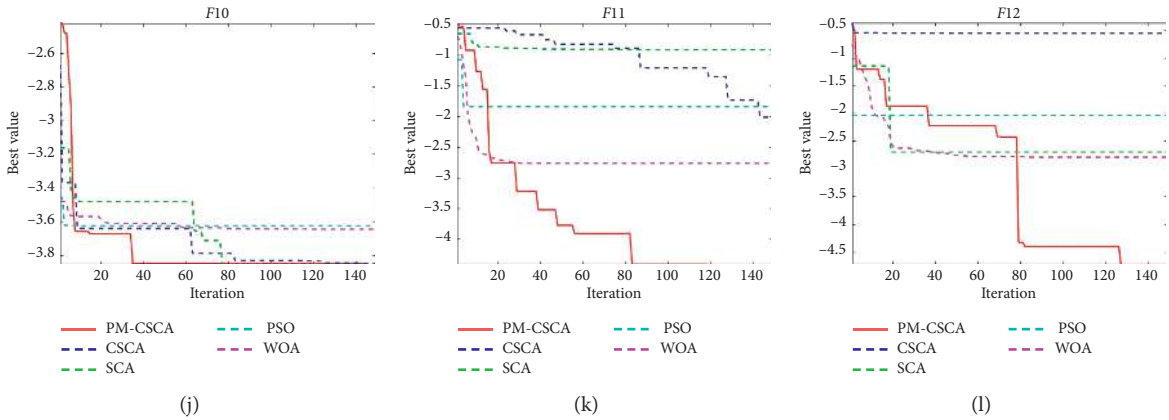


Figure 3: Convergence curves of 12 benchmark functions.

accuracy of prediction results. Obviously, the DR and ACC of intrusion detection should be high enough, while the FAR should be as low as possible. This article uses the ACC indicator as the fitness function fit(·), as shown in

$$\text{fit} = \frac{\text{TP} + \text{FN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}. \tag{18}$$

In order to verify the performance of the intrusion detection model, this paper used the NSL-KDD and UNSW-NB15 datasets commonly used in WSN intrusion detection to conduct simulation experiments. Each sample in the NSL-KDD dataset consists of 34 numerical features, 7 symbol features, and one-dimensional labels. There are five types of samples including normal data and 4 types of attack data. The four types of attacks are denial of service (DoS), sniffing (Probe), illegal access to superuser privileges by ordinary users (U2R), and illegal access from remote machines (R2L). NSL-KDD includes two training data sets (KDDTrain+, KDDTrain+_20%) and one test data set (KDDTest+). The training data set contains 21 types of attacks, and the test set adds 17 new attacks.

UNSW-NB15 is a more recent dataset than NSL-KDD, so it is more representative of real network traffic. It includes 100 GB of original network traffic and a total of 2540044 data samples. The features of this dataset are different from NSL-KDD and are more in line with the current network protocol model. It contains 10 categories, a normal category and 9 attack categories (i.e., Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worm).

Before the implementation of the algorithms, the datasets are preprocessed, including numerical, normalization, and other operations. The detection performance of five intrusion detection models was tested, respectively (SVM, kNN, PSO + kNN, SCA + kNN, and PM-CSCA + kNN). The experimental results are shown in Table 3 and the average results of 10 independent experiments are recorded. The population size of the three evolutionary algorithms of PSO, SCA, and PM-CSCA is set to 30, and the number of iterations is 120. The model PM-CSCA + kNN achieved the best results on the three indicators of ACC, DR, and FAR (indicated in bold), which means that the model can identify most WSN attack behaviors and distinguish different types.

This paper introduces evolutionary algorithms in the intrusion detection model. Figure 4 shows the iterative process of the four optimization schemes. It was found that the result of optimizing kNN by SCA is always better than that of PSO; although CSCA has a great advantage in convergence speed, the accuracy is not stable, and sometimes it will fall into the local optimum; PM-CSCA has the best optimization effect on kNN, showing strong competitiveness both in accuracy and speed.

The confusion matrix is used to evaluate the accuracy of the four detection models on NSL-KDD, as shown in Figure 5. The horizontal axis represents the predicted value, and the vertical axis represents the true value, which visually shows the misclassification of each category. It can be seen that PM-CSCA + kNN has the best detection effect.

For WSN intrusion detection systems, reducing the false alarm rate is a challenge. We conducted five independent experiments ($E1$~$E5$) on two data sets. Figure 6 Intuitively shows the comparison result of the false alarm rate of four different detection algorithms. It can be seen that the false alarm rate of PM-CSCA + kNN is extremely stable at a low level. For the convenience of showing the relationship between DR and FAR, the Receiver Operating Characteristics (ROC) curves based on two datasets are drawn, as shown in Figure 7. The ROC curves corresponding to the algorithm proposed in this article are all closest to the upper left boundary, so the effect of this prediction model is the best.

Table 3: Performance indicators comparison of five intrusion detection models (SVM, kNN, PSO + kNN, SCA + KNN, and PM-CSCA + kNN) on NSL-KDD and UNSW-NB15 datasets.

Model	NSL-KDD			UNSW-NB15		
	ACC (%)	DR (%)	FAR (%)	ACC (%)	DR (%)	FAR (%)
SVM	92.116	92.459	9.3684	92.6	91.82	8.73
kNN	94.100	95.370	8.1300	86.64	85.35	11.48
PSO + kNN	95.890	96.078	4.2105	90.64	89.86	10.08
SCA + kNN	97.952	97.321	1.6575	93.84	93.28	7.95
PM-CSCA + kNN	**99.327**	**99.206**	**0.5848**	**98.27**	**97.94**	**5.82**

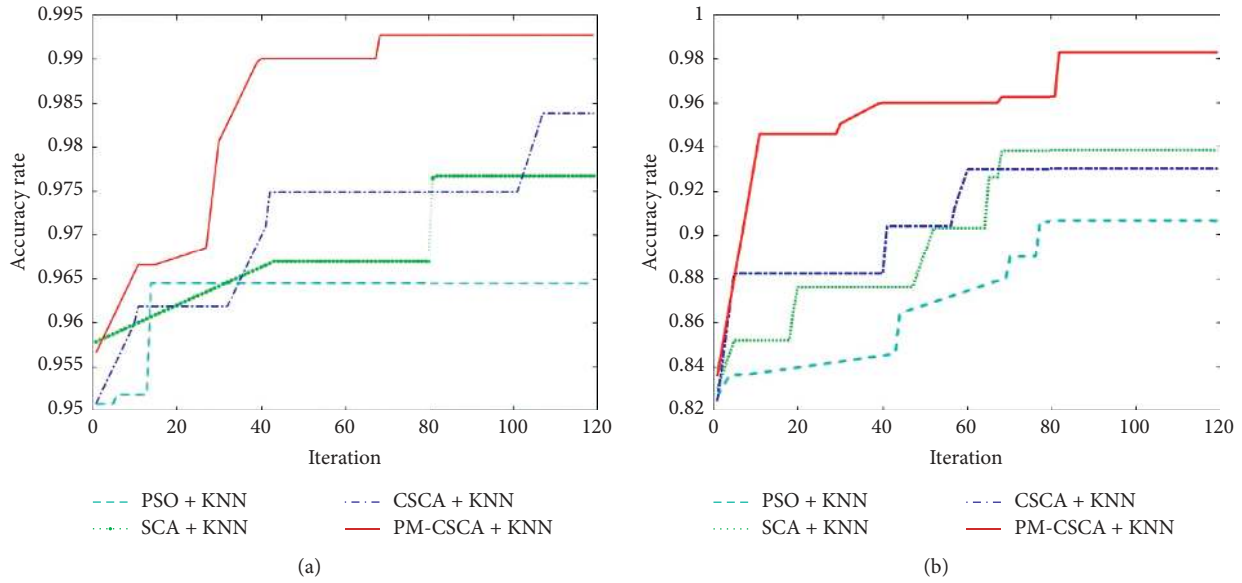


Figure 4: Comparison of the convergence curves of PSO, SCA, CSCA, and PM-CSCA. (a) Based on NSL-KDD dataset. (b) Based on UNSW-NB15 dataset.

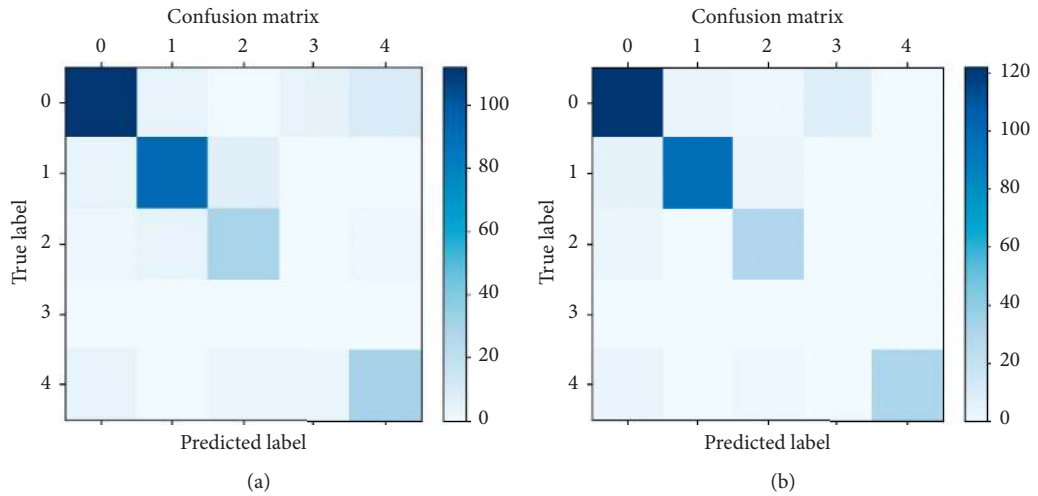
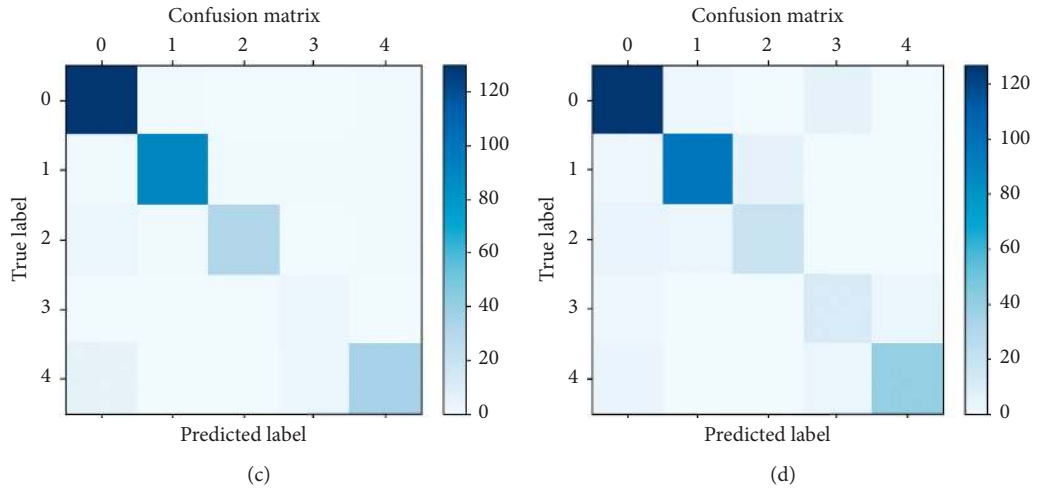
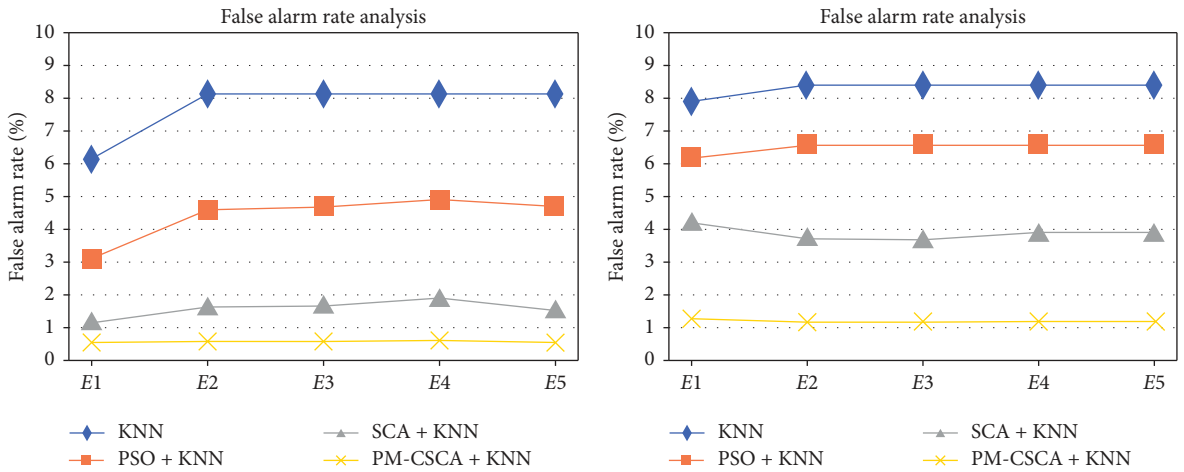


Figure 5: Continued.



(c) (d)

Figure 5: Confusion matrices of four intrusion detection models on NSL-KDD. (a) kNN. (b) PSO + kNN. (c) SCA + kNN. (d) PM-CSCA + kNN.



(a) (b)

Figure 6: Comparison of the false alarm rate of kNN, PSO + kNN, SCA + kNN, and PM-CSCA + kNN. (a) Based on NSL-KDD dataset. (b) Based on UNSW-NB15 dataset.

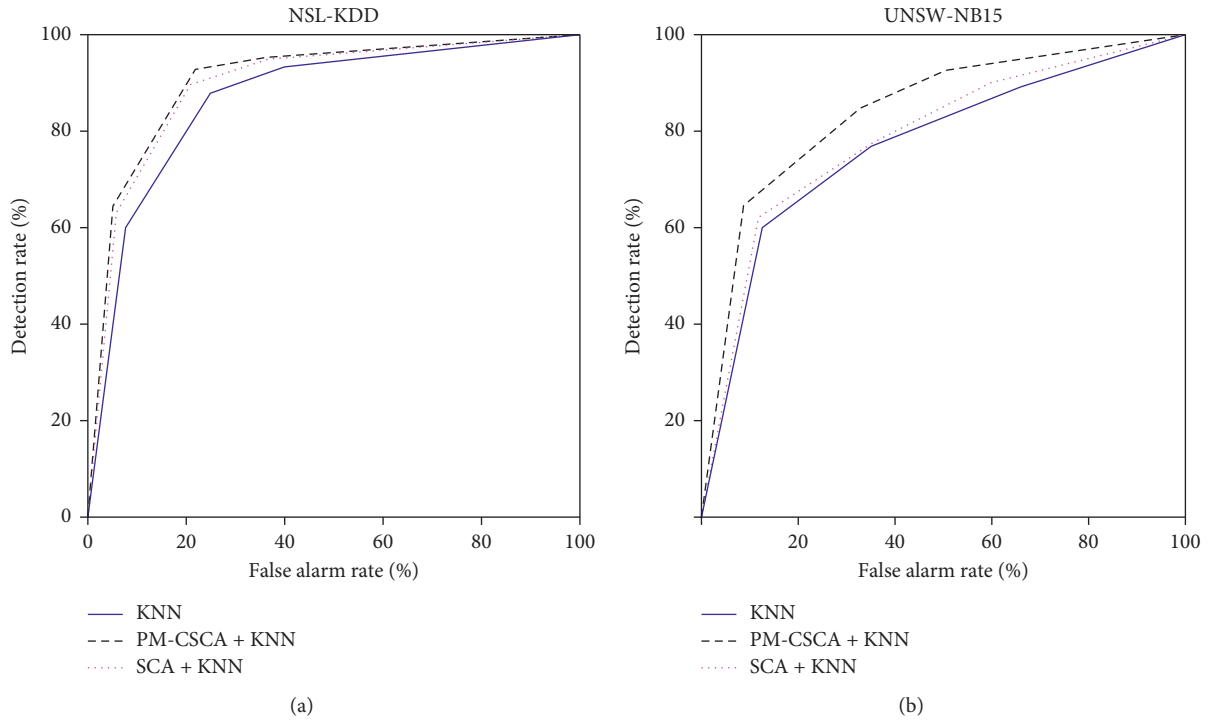


Figure 7: ROC curves of three classification algorithms on two datasets.

6. Conclusion and Future Works

Intrusion detection is one of the key issues that need to be solved urgently in practical applications of WSN. With the continuous expansion of the service area and the rapid rise of data volume, the threat and consequences of network attacks in WSN cannot be ignored. Most of the existing intrusion detection systems can only deal with specific types of attacks, and they are powerless against unknown attacks [52]. And while protecting the network security, it inevitably increases the energy consumption and transmission delay. These problems need to be paid more attention in WSN. This paper proposes a lightweight and intelligent intrusion detection model for WSN, which comprehensively considers security, energy saving, and real-time. Intelligent anomaly detection is realized through the joint use of kNN and SCA. The introduction of evolutionary algorithms makes the kNN classifier change from lazy learning to active optimization in the setting of its parameters, which significantly improves the detection accuracy. kNN and SCA are both algorithms with less computation and easy implementation, which meet the requirements of lightweight model. In order to be more efficient, this article proposes an improved version of SCA: PM-CSCA. Two technologies are integrated: compact mechanism greatly reduces the time and space required for algorithm, and PM strategy ensures the optimization accuracy, and these have been verified in tests based on benchmark functions. PM-CSCA shows good optimization ability in the benchmark function test. In simulation experiments on public data set, the intrusion detection model proposed in this paper has also been proved to be feasible and effective. In addition, the intrusion detection system for WSN is deployed in the hybrid computing mode. Cloud computing, fog computing, and AI work together to provide a feasible and efficient solution for maintaining data security in WSN.

We will do further research on the lightweight and intelligent WSN intrusion detection model, for example, how to use unsupervised machine learning techniques to deal with unpredictable cyber attacks [53]. Furthermore, more core technologies of evolutionary computing can be applied to solve big data or large-dimensional problems encountered in intrusion detection [54, 55].

The following abbreviations are used in this manuscript:

Abbreviations

WSN: Wireless sensor networks
kNN: k-nearest neighbor algorithm
SCA: Sine cosine algorithm
CSCA: Compact SCA
PM: Polymorphic mutation
AI: Artificial intelligence
IDS: Intrusion detection system
SVM: Support vector machine
PV: Perturbation vector
PDF: Probability density function
CDF: Cumulative distribution function
PSO: Particle swarm optimization
WOA: Whale optimization algorithm.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

References

[1] H. Ghayvat, S. Mukhopadhyay, X. Gui, and N. Suryadevara, "WSN- and IOT-based smart homes and their extension to smart buildings," *Sensors*, vol. 15, no. 5, pp. 10350–10379, 2015.

[2] Q.-W. Chai, S.-C. Chu, J.-S. Pan et al., "Applying adaptive and self-assessment fish migration optimization on localization of wireless sensor network on 3-D terrain," *Journal of Information Hiding and Multimedia Signal Processing*, vol. 11, no. 2, pp. 90–102, 2020.

[3] D. Ciuonzo, P. S. Rossi, and P. K. Varshney, "Distributed detection in wireless sensor networks under multiplicative fading via generalized score-tests," *IEEE Internet of Things Journal*, vol. 1, p. 1, 2021.

[4] J. Wang, Y. Gao, W. Liu, A. K. Sangaiah, and H.-J. Kim, "Energy efficient routing algorithm with mobile sink support for wireless sensor networks," *Sensors*, vol. 19, no. 7, p. 1494, 2019.

[5] B. B. Zarpelão, R. S. Miani, C. T. Kawakani, and S. C. De Alvarenga, "A survey of intrusion detection in Internet of Things," *Journal of Network and Computer Applications*, vol. 84, pp. 25–37, 2017.

[6] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: an ensemble of autoencoders for online network intrusion detection," 2018, https://arxiv.org/abs/1802.09089.

[7] I. Butuan, S. D. Merger, and R. Sankar, "A survey of intrusion detection systems in wireless sensor networks," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 266–282, 2013.

[8] T. Ma, F. Wang, J. Cheng, Y. Yu, and X. Chen, "A hybrid spectral clustering and deep neural network ensemble algorithm for intrusion detection in sensor networks," *Sensors*, vol. 16, no. 10, p. 1701, 2016.

[9] R. Vinayakumar, K. P. Soman, and P. Poornachandran, ""Applying convolutional neural network for network intrusion detection," in *Proceedings of the 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Udupi, India, December 2017.

[10] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017.

[11] K. A. P. Costa, L. A. M. Pereira, R. Y. M. Nakamura, C. R. Pereira, J. P. Papa, and A. Xavier Falcão, "A nature-inspired approach to speed up optimum-path forest clustering and its application to intrusion detection in computer networks," *Information Sciences*, vol. 294, pp. 95–108, 2015.

[12] Y. Xue, W. Jia, X. Zhao et al., "An evolutionary computation-based feature selection method for intrusion detection," *Security and Communication Networks*, vol. 2018, Article ID 2492956, 10 pages, 2018.

[13] C. Umarani and S. Kannan, "Intrusion detection system using hybrid tissue growing algorithm for wireless sensor network," *Peer-to-Peer Networking and Applications*, vol. 13, no. 3, pp. 752–761, 2019.

[14] Z. Sun, Y. Xu, G. Liang et al., "An intrusion detection model for wireless sensor networks with an improved V-detector algorithm," *IEEE Sensors Journal*, vol. 18, no. 5, pp. 1971–1984, 2017.

[15] A. A. Aburomman and M. B. Ibne Reaz, "A novel SVM-kNN-PSO ensemble method for intrusion detection system," *Applied Soft Computing*, vol. 38, pp. 360–372, 2016.

[16] M. H. Ali, B. A. D. Al Mohammed, A. Ismail, and M. F. Zolkipli, "A new intrusion detection system based on fast learning network and particle swarm optimization," *IEEE Access*, vol. 6, pp. 20255–20261, 2018.

[17] G. Bovenzi, G. Aceto, D. Ciuonzo et al., "A hierarchical hybrid intrusion detection approach in iot scenarios," 2020.

[18] P. Nancy, S. Muthurajkumar, S. Ganapathy, S. V. N. Santhosh Kumar, M. Selvi, and K. Arputharaj, "Intrusion detection using dynamic feature selection and fuzzy temporal decision tree classification for wireless sensor networks," *IET Communications*, vol. 14, no. 5, pp. 888–895, 2020.

[19] C.-M. Chen, B. Xiang, T.-Y. Wu, and K.-H. Wang, "An anonymous mutual authenticated key agreement scheme for wearable sensors in wireless body area networks," *Applied Sciences*, vol. 8, no. 7, p. 1074, 2018.

[20] T.-Y. Wu, C.-M. Chen, X. Sun, S. Liu, and J. C.-W. Lin, "A countermeasure to SQL injection attack for cloud environment," *Wireless Personal Communications*, vol. 96, no. 4, pp. 5279–5293, 2017.

[21] J. N. Chen, F. M. Zou, T. Y. Wu et al., "A new certificate-based aggregate signature scheme for wireless sensor networks," *J. Inf. Hiding Multimedia Signal Process*, vol. 9, no. 5, pp. 1264–1280, 2018.

[22] D. S. Vijayakumar and S. Ganapathy, "Machine learning approach to combat false alarms in wireless intrusion detection system," *Computer and Information Science*, vol. 11, no. 3, pp. 67–81, 2018.

[23] B. Riyaz and S. Ganapathy, "A deep learning approach for effective intrusion detection in wireless networks using CNN," *Soft Computing*, vol. 24, no. 22, pp. 17265–17278, 2020.

[24] R. Vijayanand, D. Devaraj, and B. Kannapiran, "Intrusion detection system for wireless mesh network using multiple support vector machine classifiers with genetic-algorithm-based feature selection," *Computers & Security*, vol. 77, pp. 304–314, 2018.

[25] M. Safaldin, M. Otair, and L. Abualigah, "Improved binary gray wolf optimizer and SVM for intrusion detection system in wireless sensor networks," *Journal of Ambient Intelligence and Humanized Computing*, vol. 13, pp. 1–18, 2020.

[26] Z. Nannan, W. Lifeng, Y. Jing et al., "Naive bayes bearing fault diagnosis based on enhanced independence of data," *Sensors*, vol. 18, no. 2, p. 463, 2018.

[27] S. Mirjalili, "SCA: a Sine Cosine Algorithm for solving optimization problems," *Knowledge-based Systems*, vol. 96, pp. 120–133, 2016.

[28] S. Gupta, K. Deep, S. Mirjalili, and J. H. Kim, "A modified sine cosine algorithm with novel transition parameter and mutation operator for global optimization," *Expert Systems with Applications*, vol. 154, Article ID 113395, 2020.

[29] A. Bhadoria, S. Marwaha, and V. K. Kamboj, "An optimum forceful generation scheduling and unit commitment of thermal power system using sine cosine algorithm," *Neural Computing and Applications*, vol. 32, no. 1, pp. 1–30, 2020.

[30] Y. L. Qiao, J.-S. Pan, and S. H. Sun, "Improved K nearest neighbor classification algorithm," in *Proceedings of the 2004*

IEEE Asia-Pacific Conference on Circuits and Systems, Tainan, Taiwan, December 2004.

[31] Y. Chen, X. Hu, W. Fan et al., "Fast density peak clustering for large scale data based on kNN," *Knowledge-Based Systems*, vol. 187, Article ID 104824, 2020.

[32] S. Zhang, D. Cheng, Z. Deng, M. Zong, and X. Deng, "A novel k NN algorithm with data-driven k parameter computation," *Pattern Recognition Letters*, vol. 109, pp. 44–54, 2018.

[33] S. Zhang, X. Li, M. Zong, X. Zhu et al., "Efficient knn classification with different numbers of nearest neighbors," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 5, pp. 1774–1785, 2017.

[34] Z. Deng, X. Zhu, and D. Cheng, "Efficient kNN classification algorithm for big data," *Neurocomputing*, vol. 195, pp. 143–148, 2016.

[35] H. Yang, S. Liang, J. Ni et al., "Secure and efficient kNN classification for industrial Internet of things," *IEEE Internet of Things Journal*, vol. 7, no. 11, 2020.

[36] K. Wang, X. Yu, Q. Xiong et al., "Learning to improve WLAN indoor positioning accuracy based on DBSCAN-KRF algorithm from RSS fingerprint data," *IEEE Access*, vol. 7, pp. 72308–72315, 2019.

[37] B. K. Samanthula, Y. Elmehdwi, and W. Jiang, "K-nearest neighbor classification over semantically secure encrypted relational data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 5, pp. 1261–1273, 2014.

[38] Z. Y. Meng, J.-S. Pan, and L. P. Kong, "Parameters with adaptive learning mechanism (PALM) for the enhancement of differential evolution," *Knowledge -Based Syst*, vol. 141, pp. 92–112, 2018.

[39] G. R. Harik, F. G. Lobo, and D. E. Goldberg, "The compact genetic algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 287–297, 1999.

[40] E. Mininno, F. Cupertino, and D. Naso, "Real-valued compact genetic algorithms for embedded microcontroller optimization," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 2, pp. 203–219, 2008.

[41] E. Mininno, F. Neri, F. Cupertino et al., "Compact differential evolution," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 32–54, 2010.

[42] F. Neri, E. Mininno, and G. Iacca, "Compact particle swarm optimization," *Information Sciences*, vol. 239, pp. 96–121, 2013.

[43] C. H. Chen, F. Song, F. J. Hwang et al., "A probability density function generator based on neural networks," *Physica A: Statistical Mechanics and Its Applications*, vol. 541, Article ID 123344, 2020.

[44] X. S. Xue and J. F. Chen, "Using compact evolutionary tabu search algorithm for matching sensor ontologies," *Swarm and Evolutionary Computation*, vol. 48, pp. 25–30, 2019.

[45] X. S. Xue and J. F. Chen, "Optimizing sensor ontology alignment through compact co-firefly algorithm," *Sensors*, vol. 20, no. 7, pp. 1–15, 2020.

[46] X. S. Yang and S. Deb, "Cuckoo search via lévy flights," in *Proceedings of the 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC)*, Coimbatore, India, December 2009.

[47] P. C. Song, J.-S. Pan, and S.-C. Chu, "A parallel compact cuckoo search algorithm for three-dimensional path planning," *Applied Soft Computing*, vol. 94, Article ID 106443, 2020.

[48] H. Wang, W. Wang, Z. Cui et al., "A new dynamic firefly algorithm for demand estimation of water resources," *Information Sciences*, vol. 438, pp. 95–106, 2018.

[49] X. Wang, J.-S. Pan, and S. C. Chu, "A parallel multi-verse optimizer for application in multilevel image segmentation," *IEEE Access*, vol. 8, pp. 32018–32030, 2020.

[50] J.-S. Pan, X. X. Sun, S.-C. Chu et al., "Digital watermarking with improved SMS applied for QR code," *Engineering Applications of Artificial Intelligence*, vol. 97, Article ID 104049, 2021.

[51] T. B. Jiang and S.-C. Chu, "Parallel charged system search algorithm for energy management in wireless sensor network," in *Proceedings of the 2020 2nd International Conference on Industrial Artificial Intelligence (IAI)*, Shenyang, China, October 2020.

[52] O. A. Osanaiye, A. S. Alfa, and G. P. Hancke, "Denial of service defence for resource availability in wireless sensor networks," *IEEE Access*, vol. 6, pp. 6975–7004, 2018.

[53] H. Qu, Z. Qiu, X. Tang et al., "Incorporating unsupervised learning into intrusion detection for wireless sensor networks with structural co-evolvability," *Applied Soft Computing*, vol. 71, pp. 939–951, 2018.

[54] S. F. Qin, C. L. Sun, G. C. Zhang et al., "A modified particle swarm optimization based on decomposition with different ideal points for many-objective optimization problems," *Complex & Intelligent Systems*, vol. 6, no. 2, pp. 263–274, 2020.

[55] H. Wang, M. G. Liang, C. L. Sun et al., ""Multiple-strategy learning particle swarm optimization for large-scale optimization problems," *Complex & Intelligent Systems*, vol. 23, 2020.