sensors

MDPI

Article

An Enhanced Intrusion Detection Model Based on Improved kNN in WSNs

Gaoyuan Liu [1], Huiqi Zhao [1,*], Fang Fan [1,2], Gang Liu [1], Qiang Xu [2] and Shah Nazir [3]

1 College of Intelligent Equipment, Shandong University of Science and Technology, Tai'an 271000, China; liugaoyuan@sdust.edu.cn (G.L.); fangfan@sdust.edu.cn (F.F.); lgtodd@163.com (G.L.)
2 College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao 266590, China; xuqiang@sdust.edu.cn
3 Department of Computer Science, University of Swabi, Swabi 23430, Pakistan; snshahnzr@gmail.com
* Correspondence: zhaohq@sdust.edu.cn

Abstract: Aiming at the intrusion detection problem of the wireless sensor network (WSN), considering the combined characteristics of the wireless sensor network, we consider setting up a corresponding intrusion detection system on the edge side through edge computing. An intrusion detection system (IDS), as a proactive network security protection technology, provides an effective defense system for the WSN. In this paper, we propose a WSN intelligent intrusion detection model, through the introduction of the k-Nearest Neighbor algorithm (kNN) in machine learning and the introduction of the arithmetic optimization algorithm (AOA) in evolutionary calculation, to form an edge intelligence framework that specifically performs the intrusion detection when the WSN encounters a DoS attack. In order to enhance the accuracy of the model, we use a parallel strategy to enhance the communication between the populations and use the Lévy flight strategy to adjust the optimization. The proposed PL-AOA algorithm performs well in the benchmark function test and effectively guarantees the improvement of the kNN classifier. We use Matlab2018b to conduct simulation experiments based on the WSN-DS data set and our model achieves 99% ACC, with a nearly 10% improvement compared with the original kNN when performing DoS intrusion detection. The experimental results show that the proposed intrusion detection model has good effects and practical application significance.

Keywords: wireless sensor networks; intrusion

check for updates

Citation: Liu, G.; Zhao, H.; Fan, F.; Liu, G.; Xu, Q.; Nazir, S. An Enhanced Intrusion Detection Model Based on Improved kNN in WSNs. *Sensors* **2022**, *22*, 1407. https://doi.org/10.3390/s22041407

Academic Editor: Paolo Bellavista

Received: 16 December 2021
Accepted: 15 January 2022
Published: 11 February 2022

1. Introduction

With the rapid development of real-time big data and the Internet of Things, the demand for surrounding environmental data has also increased significantly [1]. The demand for WSN products with low node costs and easy deployment will gradually expand. WSN products can break through traditional detection methods. They reduce the costs of environmental testing, and also greatly reduce the cumbersome process of traditional testing methods. As a new network, the WSN has been widely studied by scientific researchers and widely used in industry since its inception. Common applications include scenarios such as environmental detection, military operations, and information positioning. In the face of complex and changeable application scenarios, WSN nodes face various challenges, such as (1) the computing power and storage capacity of a single node is quite limited; (2) the communication capability between nodes is weak; (3) the sensor node is in a complex physical environment; (4) some mobile nodes may cause the network topology to be dynamic and random. Therefore, the security of WSN sensors is relatively low, network attacks against WSN sensors are easier, and their security problems are becoming increasingly severe.

Intrusion detection is the second line of defense for network security. An intrusion detection system (IDS) can not only resist network attacks from intruders but also strengthen the system's defense capabilities based on known attacks. From the perspective of data sources, we can divide intrusion detection systems into three categories: (1) host-based intrusion detection systems [2–4], which do not require the participation of network data, and only judge whether the data are abnormal from the intrusion detection library inside the system—this method will waste a lot of CPU resources and it is not suitable for the use of small distributed devices; (2) network-based intrusion detection systems [5,6] can acquire real-time network data packets and establish a corresponding intrusion detection library to perform pattern matching and frequency analysis and judgment on the data packets, but this method will carry large costs for the database update; (3) distributed intrusion detection systems [7–9], where the system can comprehensively consider the above two intrusion detection systems, i.e., it can not only detect the operating data of the host but also can detect network data. A WSN, as an excellent distributed device that has a low price, can be widely deployed on the edge side of the Internet of Things to protect the security of the entire Internet of Things system. Intrusion detection is classified by the detection technology, which can be divided into misuse-based intrusion detection [10] and anomaly-based intrusion detection [11,12]. The first method mainly detects data directly based on the existing signature database of the system; a problem occurs when the system's signature database is not updated in time, and thus the new intrusion behaviors cannot be detected. In the second method, the system will establish a normal working database, and then compare this with the behavior of the computer; if it does not exceed the threshold, it will be a normal behavior; otherwise, the system will activate an alarm. This method is also the most common intrusion detection method, but this method often encounters challenges wherein the non-attack behavior is not within the scope of the normal working mode; the false positive rate of this method is relatively high. However, the application of intrusion detection systems on WSNs presents a great challenge. Because WSNs cannot provide enough information required by traditional intrusion detection technology, traditional intrusion detection technology cannot be directly applied to WSNs. Therefore, building a simple and lightweight WSN intrusion detection technology has become an important issue in the field of WSN security.

Intrusion detection technology is an important technique to ensure product security. Therefore, it is very important to accurately identify various attacks in the network. At present, there are some mature WSN intrusion detection systems based on convolutional neural networks [13], random forest [14], naive Bayes [15], decision tree [16], and other intrusion detection systems based on machine learning. We know that distributed equipment has extremely high requirements regarding space complexity. The space complexity of a convolutional neural network is $S\left(k^2 \times Cout \times Cin\right)$, where k is the kernel size; the space complexity of random forest is $S(N + D \times Split \times TreeNum)$, where N is the number of samples and D is the number of features; and the space complexity of naive Bayes is $S(N \times D)$, but naive Bayes must ensure that the samples are independent of each other to ensure a good effect. kNN [17], with almost the same space complexity as Naive Bayes, is also an excellent machine learning method, whose space complexity is $S(N \times D)$. The k-Nearest Neighbor (kNN) method is a normal machine learning method, whose structure is simple and easy to implement, and the classification effect is good. Because it eliminates the training process of neural networks, it is often used as a lightweight machine learning method in the intrusion detection systems of traditional networks. Compared with traditional networks, due to the small size of the WSN sensor and low computing power, purely embedding the artificial intelligence model into the WSN sensor may cause slower operation and low performance. Therefore, we can use edge nodes close to the WSN sensor nodes to train and deploy machine learning models at the edge of the network, closer to users and data sources, in an edge-wise manner [18–20]. There are two problems when we want to use kNN: one is the measurement of distance, and the other is the selection of the k parameter. For the distance measurement problem, kNN needs to calculate the distance between each sample size of the test set and the training set, which means that it is challenging for kNN to process larger data in reality. Tan et al. [21] used a different strategy to refine the incoming data and greatly improve the classification effect of kNN; for the selection of the k value, Liang [22] and others have made some progress. Huang uses a weighted method to improve kNN. However, these works have not simultaneously optimized the k and weighted above-mentioned problems.

Optimization is one of the most commonly used methods to solve complex problems. For kNN distance selection and K value selection, many scholars also use optimization algorithms to solve them. For example, Chen [23] used the PSO algorithm to optimize kNN weights. Xu [24] optimized the kNN distance formula through GWO [25], and specifically solved the kNN distance selection problem; Tahir [26] et al. optimized the k value selection of kNN through TS [27], and specifically solved the K value selection problem. Of course, common optimization algorithms include evolutionary-inspired algorithms (GA) [28], differential evolution algorithm (DE) [29], whale algorithm (WOA) [30], cat colony algorithm (CA) [31], multiverse algorithm (MVO) [32], and quasi-affine optimization algorithm (QUATER) [33] inspired by natural physics. For WSN nodes with weak computing power, overly complex optimization algorithms cannot be selected for optimization. Therefore, choosing a lightweight and high-performance optimization algorithm is particularly meaningful in the process of WSN intrusion detection. The Arithmetic Optimization Algorithm (AOA) is a new stochastic optimization algorithm proposed by Australian scholar Mirjalili in 2020 [34]. The algorithm has few control parameters, a simple structure, ease of implementation, and has an excellent performance in a variety of industrial optimization problems. In this regard, this article will focus on using the AOA algorithm to optimize the kNN weight and k value selection. However, considering the no free lunch theorem [35], this paper will also compare the kNN optimized by other optimization algorithms with the improved version of the AOA algorithm that we propose.

In this article, we mainly offer the following contributions:

- We propose an intelligent intrusion detection model based on edge intelligence that is deployed at the edge of the WSN node ($\mathrm{kNN}_{\mathrm{PL-AOA}}$);
- We propose a parallelized arithmetic optimization algorithm and achieve outstanding results compared to another algorithm;
- Through standard data set testing, our edge intelligent intrusion detection model has good performance in detecting DoS attacks.

The main structure of the article is as follows:

In Section 2, we describe related works to introduce AOA and kNN, which we use in this article. In Section 3, we introduce the improved version of AOA proposed in this article in detail and compare it with other algorithms; Section 4 introduces the intrusion detection system based on WSN, which includes the improved kNN formula and the method of combining PL-AOA and kNN; Section 5 describes the simulation experiment and data statistics of WSN intrusion detection; Section 6 provides the conclusions and future work.

2. Related Works

2.1. Arithmetic Optimization Algorithm (AOA)

The Arithmetic Optimization Algorithm (AOA) is a new type of swarm intelligence optimization algorithm proposed by Mirjalili in 2020 The algorithm has a simple structure, fewer parameters, and is easy to implement. Its search process is mainly controlled by basic mathematical operators, namely multiplication (M "×"), division (D "÷"), subtraction (S "−"), and addition (A "+")).

Arithmetic Optimization Algorithms are used to solve optimization problems. First, AOA is achieved by creating multiple initial random candidate solutions $X \in (x_1, x_2, x_3, \ldots, x_n)$. After the AOA algorithm is initialized, it will first enter the exploration stage. Before the exploration stage, Math Optimizer Accelerated (MOA) is calculated. MOA is obtained by the following Equation (1):

$$MOA(C\_iter) = Min + C\_iter \times \left(\frac{Max - Min}{M\_Iter}\right) \tag{1}$$

where C_Iter represents the current iteration $C\_Iter \in (1, M\_Iter)$. Max and Min respectively represent the maximum and minimum values of the acceleration function. $MOA(C\_Iter)$ represents the function value in the tth iteration, which is obtained by Equation (2):

$$x_{ij}(C_{Iter}+1) = \begin{cases} best(x_j) \div (MOP+\epsilon) \times ((ub_j - lb_j) \times \mu + lb_j), r_2 < 0.5 \\ best(x_j) \times MOP \times ((ub_j - lb_j) \times \mu + lb_j) \quad , r_2 \geq 0.5 \end{cases} \tag{2}$$

where $x_i(C\_Iter+1)$ represents the i-th solution in the next iteration, $x_{ij}(C\_Iter)$ represents the j-th position of the i-th solution in the current iteration, $best(x_j)$ is the j-th position in the best iteration, ϵ is a small integer number, ub_j and lb_j represent the upper limit and lower limit of the j-th position, respectively, $\mu = 0.5$.

According to the arithmetic operators, using the division (D) operator or the multiplication (M) operator can obtain highly distributed values or decisions. These will help the algorithm exploration mechanism, by using multiplication (M) or division (D). Due to the exploration mechanism of AOA, it is possible to find the approximate optimal solution through multiple iterative explorations in the solution space, thereby providing the possibility to obtain a more promising optimal solution in the subsequent optimization phase (development phase). MOP is a mathematical optimizer; $MOP(C\_Iter)$ represents the function value of the t-th iteration, as shown in Equations (3) and (4):

$$MO\ P(C\_Iter) = 1 - \frac{C\_Iter^{\frac{1}{\alpha}}}{M\_Iter^{\frac{1}{\alpha}}} \tag{3}$$

where C_Iter represents the current iteration, and (M_Iter) represents the maximum number of iterations. α is a sensitive parameter, which defines the development accuracy in the iterative process; in this paper, $\alpha = 5$.

$$x_{ij}(C_{Iter}+1) = \begin{cases} best(x_j) + MOP \times ((ub_j - lb_j) \times \mu + lb_j) \quad , r_3 \geq 0.5 \\ best(x_j) - MOP \times ((ub_j - lb_j) \times \mu + lb_j) \quad , r_3 < 0.5 \end{cases} \tag{4}$$

In order to effectively balance the exploration and development stage of the algorithm, r_1 is used in AOA as the condition for the algorithm to transition from exploration to development. r_1 is a random number $[0,1]$. When $r_1 > MOA$, the candidate solution tries to approximate the most optimal solution and diverges when the algorithm in the exploratory stage. When $r_1 \leq MOA$, the candidate solution tends to approximate the optimal solution when the algorithm in the development stage.

2.2. K-Nearest Neighbor (kNN)

The K-Nearest Neighbor (kNN) algorithm is a theoretically mature method and one of the most commonly used machine learning algorithms, which is widely used in various practical problems [36–38]. The basic idea of the kNN algorithm is that, in the feature space, if most of the k-nearest samples near a sample belong to a certain category, the sample also belongs to this category. In kNN, when a new instance appears, the k-nearest instance is directly found in the training data set, and this new instance is assigned to the class with the largest number of instances among the k training instances. Of course, for kNN, the classification result is affected by three factors: k value setting, distance measurement method, and decision rules. Among them, decision rules often follow "the minority obeys the majority". For optimization, the focus is often on optimizing the k value and distance test.

The k value is the only parameter of the kNN algorithm. The selection of the k value has a very important impact on the prediction results of kNN. A small k value will lead to a large prediction error and even noise. A large k value will cause underfitting. Under these circumstances, the forecast model is too simple. For the distance measurement of spatial samples, it is also a crucial factor that affects kNN prediction. Commonly used distance measurement methods include Euclidean distance, Mahalanobis distance, angle cosine distance, etc. In the calculation process, the smaller the distance between the two samples, the closer similarity of the two samples; otherwise, the similarity of the two samples is not good.

Suppose that there are two samples x_i and x_j in the D dimensional feature; we easily formulate the samples as: $x_i = (x_{i1},\ x_{i2}, \ldots,\ x_{iD})$ and $x_j = \left(x_{j1},\ x_{j2}, \ldots,\ x_{jD}\right)$. The distance between two samples is $d\left(x_i,\ x_j\right)$. Normally, kNN uses Euclidean distance to measure the similarity between samples, as shown in Equation (5):

$$d\left(x_i, x_j\right) = \sqrt{\sum_{k=1}^{D}\left(x_{ik} - x_{jk}\right)^2} \tag{5}$$

3. Improved AOA

In this section, we explain in detail the main ideas and implementation schemes of PL-AOA. We show that it can overcome the problem of AOA, which has slow convergence speed and prematurely falls into the local optimum when solving complex problems.

3.1. Lévy AOA

Lévy Flight

Compared with Gaussian mutation, Lévy flight [39] random walk is a better search strategy; its flight step size satisfies a stable distribution with heavy tails. By using Lévy flight in the optimization algorithm, we can expand the search range, increase the diversity of the population, and make it easier to jump out of local convergence. The updated formula of Lévy flight is given in Equation (6):

$$x_i^{t+1} = x_i^t + \alpha \times L\acute{e}vy(\lambda) \tag{6}$$

where x_i^t represents the position of x_i and the t-th generation, $\times$ is the point multiplication operation, α is the Levy flight step control amount, $Levy(\lambda)$ is the random search path, and $Levy\ \sim\ u = t(-\lambda),\ \Lambda \in (1,3]$.

In this section, we effectively combine Lévy flight with the AOA algorithm. We add Lévy flight to the iterative process of the AOA algorithm, which enhances the population diversity of the algorithm, to improve the problem of slow convergence of the AOA algorithm and solve complex problems wherein the system prematurely falls into the problem of local optima. After the AOA algorithm updates the solution, the Lévy flight leads the solution to a new $x\prime$ as shown in Equation (7):

$$x'_{ij} = x_{ij} + \alpha \times L\acute{e}vy(\lambda) \tag{7}$$

3.2. Parallel Lévy AOA (PL-AOA)

Parallel Strategy Based on Lévy AOA

In order to solve this problem, we adopt a parallel strategy to compensate for the negative effects of Lévy flight in order to enhance the communication capabilities between the populations. Normally, this method does not increase the complexity of the algorithm. The population size is taken as 40, divided into four groups [40], and the algorithm steps are as follows:

(1) Initialization:

Initialize the population P and related parameters, and divide P into 4 groups: $g_i\ (i = 1, 2, 3, 4)$.

(2) Evaluation:

Evaluate the fitness value of each particle in the population.

(3) Update:

Use Equations (2) and (4) to update the position of each particle. Then, update each group's historical optimal *pbest* and global optimal $gbest_i$ $(i = 1, 2, 3, 4)$.

(4) Communication:

Communicate between groups every k generations: choose the best solution in the group for Lévy flight.

(5) Termination:

Repeat steps 2 to 5. If the predefined function value has been obtained or all iterations have been completed, record the global optimal particle and its fitness value until the end of the optimization process.

In addition, the flowchart and pseudo code of the algorithm are shown in Figure 1 and Algorithm 1.

Algorithm 1 Pseudocode of PL-AOA

```
1: Initialize the parameters related to the algorithm: $ub, lb, Dim, max\_itergroup = 4$.
2: Generate initial population $X$ containing $N$ individuals $X_i(i = 0, 1, 2, 3, \cdots, N)$.
3: Divide $X$ into 4 groups.
4: Do
5:    if $r_1 > MOA$
6:            Update the $X$ by Equation (1).
7:     else
8:            Update the $X$ by Equation (2).
9:     for i = 1:$group$
10:            for i = 1:$Dim$
11:               if $f_{winner} < f_{gbest}$
12:                  Update the best solution obtained so far.
13:                  Change flight status according to iteration.
14:               end
15:               if $iteration = 50$
16:                  Update the $pbest$ by Equation (6) and calculate its fitness value.
17:               if $f_{gbest} < f_{pbest'}$
18:                  Update the best solution obtained so far.
19:                  Change flight status according to iteration.
20:         end
21:      end
22: While $(t < max\_iter) or\ (get\ the\ expected\ function\ value)$.
23: Return the best solution obtained so far as the global optimum.
```

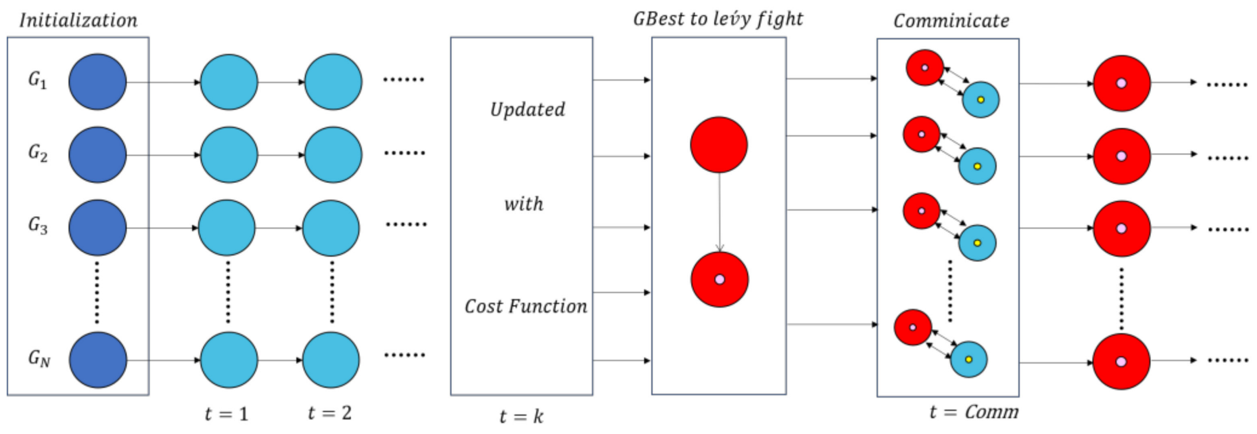

Figure 1. The PL-AOA algorithm flow.

4. An Edge-Intelligent WSN Intrusion Detection System

4.1. Weighted kNN

Though the application of Euclidean distance is more common, it is suitable for situations where the metric standards of each component of the sample vector are uniform. For most statistical problems, since the values of the sample components contribute equally to the Euclidean distance, satisfactory results are often not achieved. Especially when the dimensional difference in the fluctuation range of each component is large, this will cause the contribution of each component to the overall difference to be large, and even the contribution of a certain sample will be almost negligible. When each component is a quantity of a different nature, the size of the Euclidean distance is related to the unit of the sample component. In practical applications, it is impossible to guarantee that the value of a certain dimension has a small fluctuation range, so we can consider weighting $s_k(k = 1, 2, \ldots, D)$ for each coordinate sub-scalar. In other words, the coordinates with large changes have smaller weight coefficients than those with small changes, and the differences between different attributes of the samples are quantified to the same interval. This is the method of standardized Euclidean distance, and the formula is as follows in Equation (8):

$$d(x_i, x_j) = \sqrt{\sum_{k=1}^{D} \frac{\left(x_{ik} - x_{jk}\right)^2}{s_k}} \tag{8}$$

As a commonly used machine learning algorithm, many scholars have also improved it, mainly around the adjustment of the k parameter [41,42]. However, no in-depth research has been performed on the determination of k, selection of distance functions, or setting of distance weights. For all sample-based distributions, data characteristics, and analysis requirements, they can all be regarded as typical optimization problems. With the help of the optimization ability of a meta-heuristic algorithm, we can obtain a more reasonable and effective kNN classification model.

4.2. PL-AOA Combined with kNN

The kNN parameter k and the distance weight s_i largely determine the classification effect. However, these two parameters are often considered to be determined under normal circumstances. The PL-AOA proposed in this article can be used to optimize the relevant parameters of kNN to obtain the best or near-best classification results.

The samples in the D dimensional feature space correspond to the N solution vectors $X_i(i = 1 \ldots N)$ of the evolutionary algorithm. The first dimension is the parameter k of kNN, which can be set as a random integer within a certain range as required, but the value of k must be a positive integer. $s_{ij} \in [0, 1]$, which is a random number, represents the j-th distance weight in the i-th solution. The evolutionary algorithm will continuously search and iterate under the guidance of the fitness function, and finally output the optimal solution or approximate optimal value [43], which is the most suitable kNN-related parameter. In order to demonstrate our model more clearly, the model's structure is shown in Figure 2.

4.3. WSN Intrusion Detection System

The wireless sensor network (WSN) has become an increasingly important research area due to its wide range of real-time applications such as critical military surveillance, battlefields, building security surveillance, forest fire surveillance, and medical care [44]. A WSN consists of a large number of autonomous sensor nodes, which are distributed in different areas of interest to collect important data and wirelessly transmit the collected data to more powerful nodes, called sink nodes or base stations (BS) [45]. The data transmitted across the network depend on the dedicated WSN protocol, but the WSN is extremely vulnerable to attacks due to its open and distributed nature and the limited resources of sensor nodes.

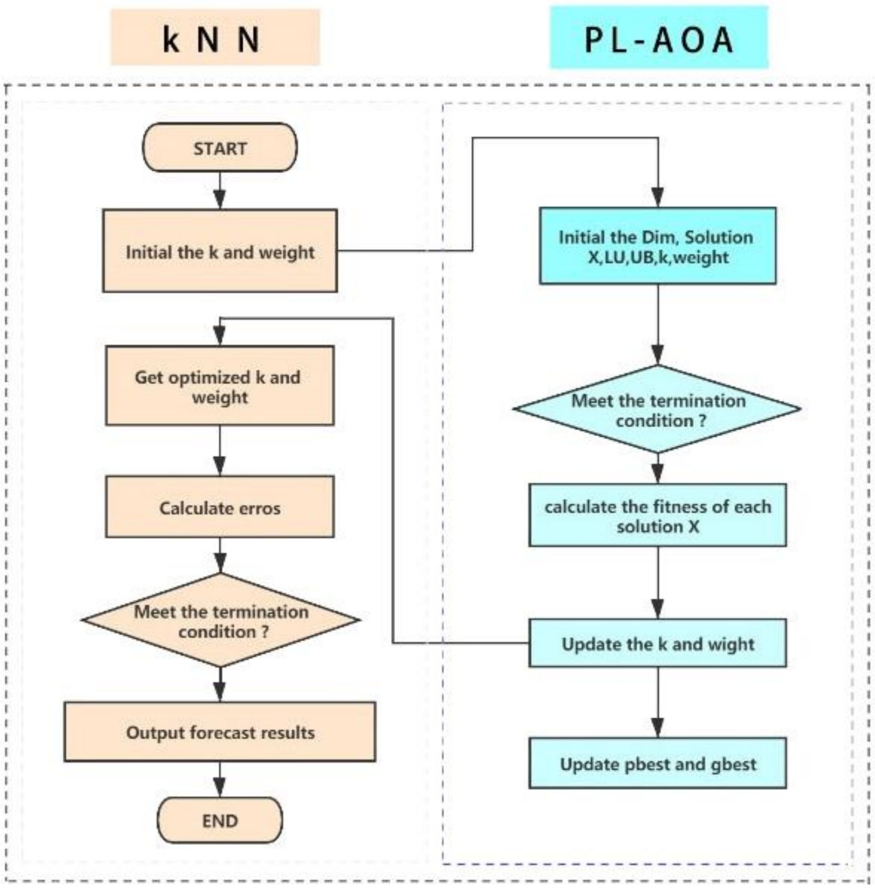
k N N
PL-AOA
START
Initial the k and weight
Get optimized k and weight
Calculate erros
Meet the termination condition ?
Output forecast results
END
Initial the Dim, Solution X,LU,UB,k,weight
Meet the termination condition ?
calculate the fitness of each solution X
Update the k and wight
Update pbest and gbest

Figure 2. PL-AOA combined with kNN.

Since the process of avoiding or preventing security threats is not always successfully completed, an intrusion detection system (IDS) is needed to detect known and unknown attacks and send out alerts about them to sensor nodes [46]. An IDS can detect suspicious or abnormal activity and trigger an alarm when an intrusion occurs. The realization of a WSN IDS is more challenging than other systems, because sensor nodes are usually designed to be small and cheap, and they do not have enough hardware resources. In addition, there is no dedicated data set that contains general configuration files and attacks in WSN that can be used to detect attackers' signatures. Considering the above challenges, we deploy edge nodes outside the WSN to enable the WSN to establish communication with the edge nodes, and we use edge intelligence to facilitate WSN intrusion detection. Figure 3 shows the edge-intelligent WSN intrusion detection system.

4.4. Performance Evaluation of Intrusion Detection System

Machine learning usually uses the following four criteria to evaluate the performance of a model: true positive (TP), true negative (TN), false positive (FP), and false negative (FN). They are also used to calculate various performance evaluation indicators, such as detection rate (DR), false positive rate (FPR), and accuracy rate (ACC). The calculation method is as follows:

$$DR = TP/(TP + FN) \tag{9}$$

$$FPR = FP/(FP + TN) \tag{10}$$

$$ACC = (TP + TN)/(TP + FN + FP + TN) \tag{11}$$

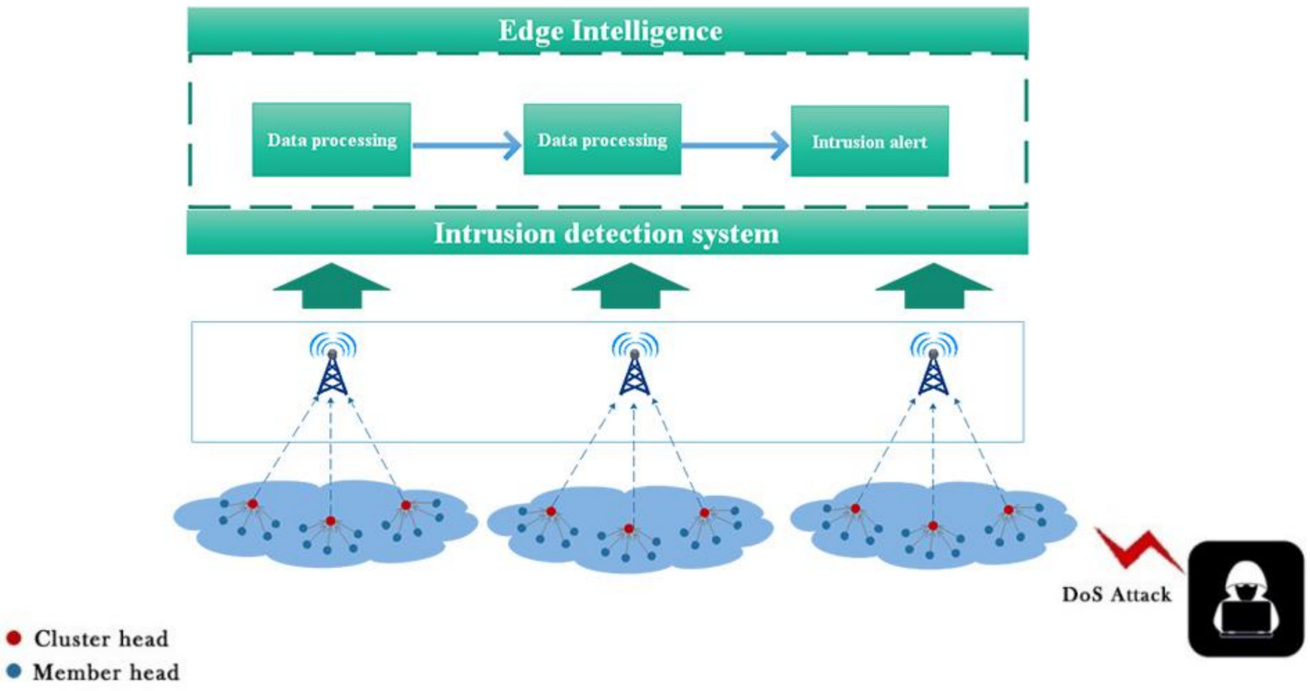

Figure 3. The model WSN intrusion detection system.

Among them, *DR* represents the probability of a positive prediction in a sample with a normal actual value. *FPR* is the probability of a positive prediction in a sample with an abnormal actual value. *ACC* divides the number of samples correctly predicted by the total number of samples to indicate the accuracy of the prediction results. This article uses the *ACC* indicator as a fitness function, and its formula is as follows:

$$fitness = (TP + TN)/(TP + FN + FP + TN) \tag{12}$$

5. Simulation Experiment and Analysis

5.1. The Experimental Results and Conclusions of PL-AOA

As described in this section, in order to test the performance of PL-AOA, we used the original standard test function of AOA to conduct comparative experiments on PL-AOA, AOA, SCA [47], and MVO. This is a complex problem, so we designed the experiment as follows, focusing on testing the optimization of PL-AOA for complex functions. The test functions that we used included two single-mode functions ($f_1 \sim f_2$), four multi-mode functions. ($f_3 \sim f_6$), and six complex functions ($f_7 \sim f_{12}$), as shown in Table 1. The test convergence curve is shown in Figure 4, and the test results are shown in Table 2.

Table 1. The 12 benchmark functions.

Function	Dim	Range	F_{min}
$f_1(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	$[-100, +100]$	0
$f_2(x) = \sum_{i=1}^{n} \left(\sum_{j-1}^{i} x_j \right)^2$	30	$[-100, +100]$	0
$f_3(x) = \sum_{i=1}^{n} -x_i \sin\left(\sqrt{\lvert x_i \rvert} \right)$	30	$[-1.28, +1.28]$	0
$f_4(x) = \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) + 10 \right]$	30	$[5.12, +5.12]$	0
$f_5(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n} \sum_{i=1}^{n} x_i^2} \right) - \exp\left(\frac{1}{n} \sum_{i=1}^{n} \cos(2\pi x_i) \right) + 20 + e$	30	$[-32, +32]$	0
$f_6(x) = \frac{1}{4000} \sum_{i=1}^{\pi} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}} \right) + 1$	30	$[-5.12, +5.12]$	0
$f_7(x) = \left(\frac{1}{500} * \sum_{i=1}^{25} \frac{1}{i + \sum_{i=1}^{2} (x_j - x_{ij})} \right)$	30	$[-65, 65]$	0
$f_8 = 4 * x_1^2 - 2.1 * \frac{x_1^6}{3 + x_1 * x_2} - 4 * x_2^2 + 4 * x_2^4$	2	$[-5, +5]$	0
$f_9(x) = -\frac{1 + \cos\left(12\sqrt{x_1^2 + x_2^2} \right)}{0.5\left(x_1^2 + x_2^2 \right) + 2}$	2	$[-2, +2]$	3
$f_{10}(x) = -\sum_{i=1}^{4} \left[(X - a_i)(X - a_i)^T + c_i \right]^{-1}$	4	$[-10, +10]$	-10.1532
$f_{11}(x) = -\sum_{i=1}^{7} \left[(X - a_i)(X - a_i)^T + c_i \right]^{-1}$	4	$[-10, +10]$	-10.4028
$f_{12}(x) = -\sum_{i=1}^{10} \left[(X - a_i)(X - a_i)^T + c_i \right]^{-1}$	4	$[-10, +10]$	-10.5363

Table 2. Comparison results of PL-AOA, AOA, SCA, and MVO on 12 benchmark functions.

Function	Algorithm	Best Value	AVG	STD
f_1	PL-AOA	0	0	0
	AOA	0	0	0
	SCA	2.09×10^{-14}	9.71085×10^{-9}	9.10029×10^{-9}
	MVO	0.013127	0.0473636	0.025148762
f_2	PL-AOA	0	0	0
	AOA	3.39×10^{-5}	6.7766×10^{-6}	1.35532×10^{-5}
	SCA	3.35×10^{-6}	0.84790582	1.01309437
	MVO	0.23415	0.25472	0.167577958
f_3	PL-AOA	−3251.961	−3072.60658	192.0813616
	AOA	−2669.3232	−2828.61848	127.3300739
	SCA	−2060.5021	−2206.09518	222.1806306
	MVO	−2511.7046	−2889.61756	225.6564613
f_4	PL-AOA	0	0	0
	AOA	1.85×10^{-8}	9.6634×10^{-14}	1.93268×10^{-13}
	SCA	45.627	2.95231933	5.879472705
	MVO	9.9668	13.93912	4.664720195

Table 2. *Cont.*

Function	Algorithm	Best Value	AVG	STD
f_5	PL-AOA	8.88×10^{-16}	8.88×10^{-16}	0
	AOA	1.62×10^{-8}	3.25×10^{-9}	6.4912×10^{-9}
	SCA	9.33×10^{-8}	3.45×10^{-7}	2.14341×10^{-7}
	MVO	0.040718	5.09×10^{-2}	0.016216161
f_6	PL-AOA	0	2.14×10^{-12}	3.96128×10^{-13}
	AOA	2.34×10^{-14}	1.98×10^{-10}	8.2776×10^{-14}
	SCA	0.0033891	1.26×10^{-1}	0.177027801
	MVO	0.20291	3.81×10^{-1}	0.134430398
f_7	PL-AOA	0.998	0.998	0
	AOA	7.874	2.377862	2.295234559
	SCA	0.99801	3.744742	3.61965409
	MVO	0.998	0.998	0
f_8	PL-AOA	−1.0316	−1.0316	0
	AOA	−1.0315	−1.03152	9.79796×10^{-5}
	SCA	−1.0314	−1.03124	0.000621611
	MVO	−1.0316	−1.02918	0.00459147
f_9	PL-AOA	−3.859	−3.85222	0.00484124
	AOA	−3.8549	−3.85486	0.003883607
	SCA	−3.8503	−3.85166	0.001654811
	MVO	−3.8628	−3.8628	0
f_{10}	PL-AOA	−5.0579	−5.04412	0.018240329
	AOA	−3.6494	−3.09772	0.96147737
	SCA	−2.8665	−2.53446	0.823849153
	MVO	−2.6828	−1.843174	1.282266985
f_{11}	PL-AOA	−7.6701	−7.3831	0.826224943
	AOA	−2.9294	−2.30354	1.040143612
	SCA	−3.0656	−2.12771	2.042497321
	MVO	−2.7659	−1.87218	3.060140019
f_{12}	PL-AOA	−5.7896	−4.91552	1.09829503
	AOA	−2.4736	−2.14244	1.972803802
	SCA	−4.699	−3.916174	1.515732845
	MVO	−2.4273	−1.36822	3.881249547
Compared with the four algorithms	Algorithm	Win	Win	Win
	PL-AOA	9	9	8
	AOA	0	0	1
	SCA	0	0	0
	MVO	1	1	1

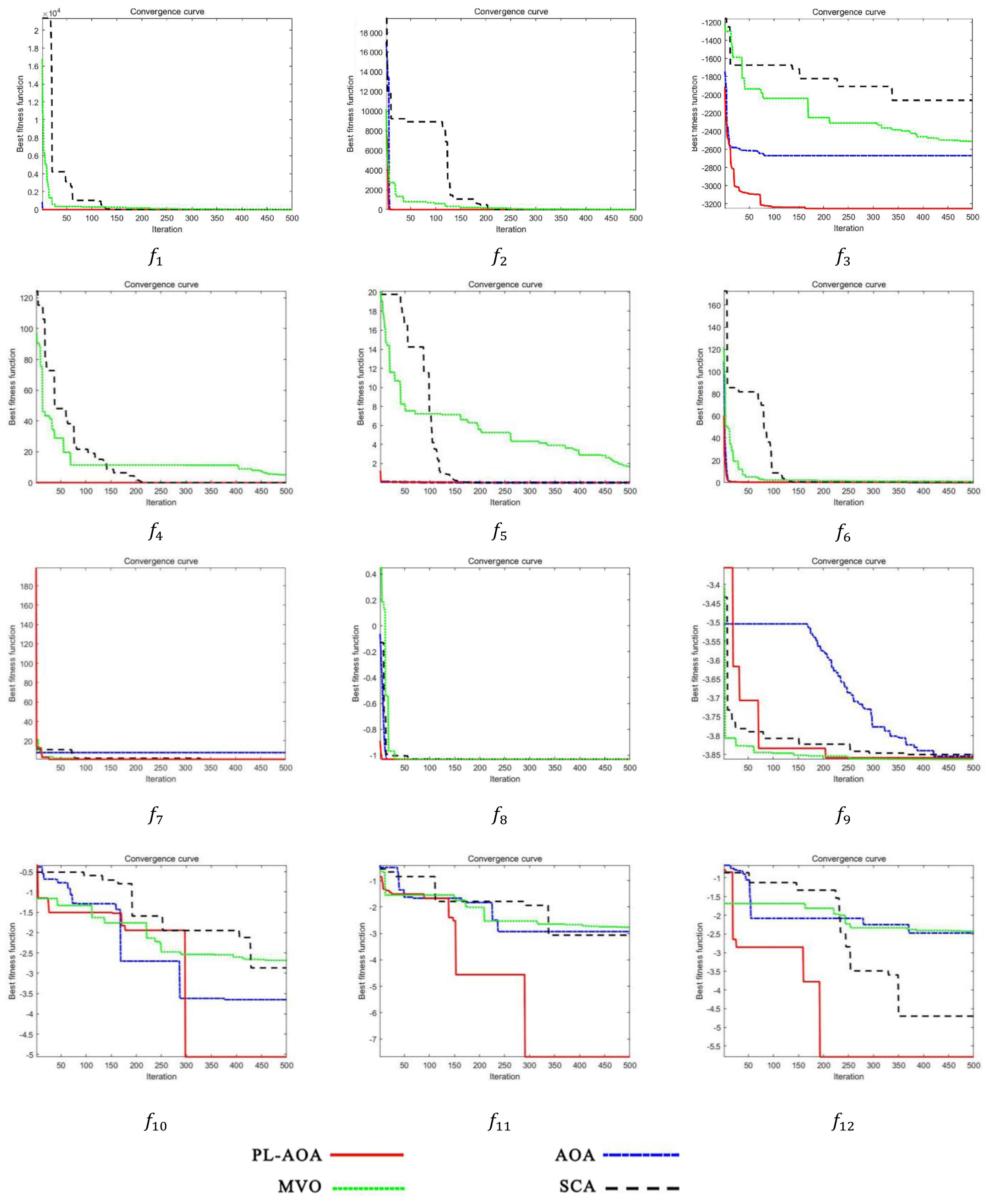

Figure 4. The convergence curves of test function.

Through three sets of benchmark function tests, it can be seen that PL-AOA has achieved the absolute advantage in algorithm comparison, and its performance in complex functions is more prominent. The numbers in bold indicate the best results for each set of tests which PL-AOA has achieved the nine best results in the 12 benchmark tests. In the standard deviation comparison, the first results were achieved eight times, which proves that the PL-AOA proposed in this paper has good optimization strength and reliable stability. Moreover, the average performance was improved by nearly 80% compared with the original AOA.

5.2. The Experimental Results and Conclusions of WSN Intrusion Detection System

In order to verify the practicability of the intrusion detection model proposed in this paper, the WSN intrusion detection data set WSN-DS [48] was used in simulation experiments. WSN-DS collects data from the network simulator 2 (NS-2), and then processes it to generate a data set of 23 features. Four types of DoS attacks are defined in WSN-DS: Blackhole, Grayhole, Flooding, and Scheduling attacks. The data distribution is shown in Table 3.

Table 3. The data of WSN-DS.

Data Set	**The Type of Data**				
	Normal	**Blackhole**	**Grayhole**	**Flooding**	**Scheduling Attacks**
Number	340,066	10,049	14,596	3312	6638

Before executing the algorithm, the data set was preprocessed, including numerical values, normalization, and other operations. The detection performance of four intrusion detection models (kNN, kNN_{PSO}, kNN_{AOA}, kNN_{PL-AOA}) was tested. The experimental results are shown in Table 4, and the average results of 30 independent experiments were recorded. The total population of the three evolutionary algorithms of PSO, AOA, and PL-AOA was set to 20, and the number of iterations was 100.We can clearly find the model kNN_{PL-AOA} achieved the best results on the three indicators of ACC, DR, and FPR. This shows that the model can identify most of the DoS attacks that affect a WSN and can distinguish different types.

Table 4. Classification effects of the four models.

Model	**ACC (%)**	**DR (%)**	**FPR (%)**
kNN	0.91162	0.95291	0.51429
kNN_{PSO}	0.92893	0.94226	0.035714
kNN_{AOA}	0.97727	0.97861	0.045455
kNN_{PL-AOA}	0.99721	0.99171	0.068966

In machine learning, the confusion matrix can be used to evaluate the accuracy of the four detection models. Here, we prove the accuracy of the four models by drawing the confusion matrix, as shown in Figure 5. The horizontal axis represents the predicted value, and the vertical axis represents the true value, visually showing the misclassification of each category. It can be seen that the kNN_{PL-AOA} model proposed in this paper has the best detection effect.

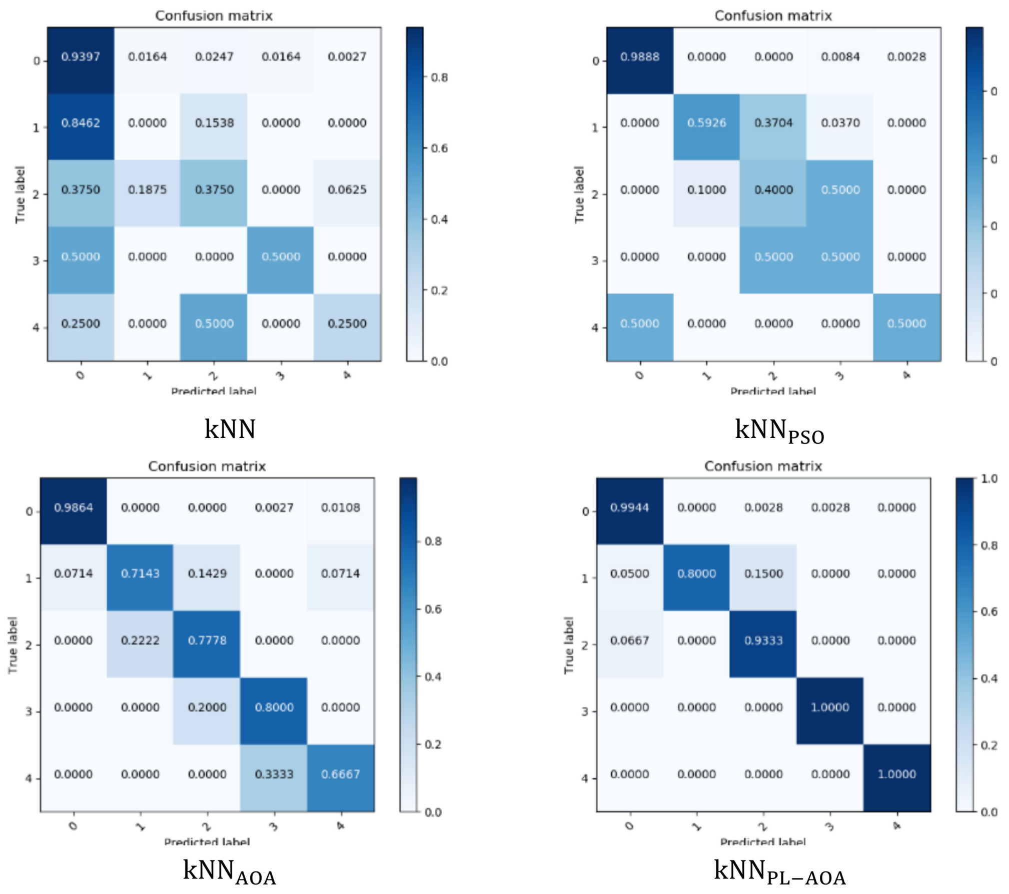

Figure 5. kNN, kNN$_{PSO}$, kNN$_{AOA}$, and kNN$_{PL-AOA}$ confusion matrix.

For WSN intrusion detection systems, reducing the false positive rate is a challenge. We conducted five independent experiments using the data set. Figure 6 visually shows the comparison results of the false positive rate of the four different detection algorithms. It can be seen that the false positive rate of kNN$_{PL-AOA}$ is extremely stable at a low level.

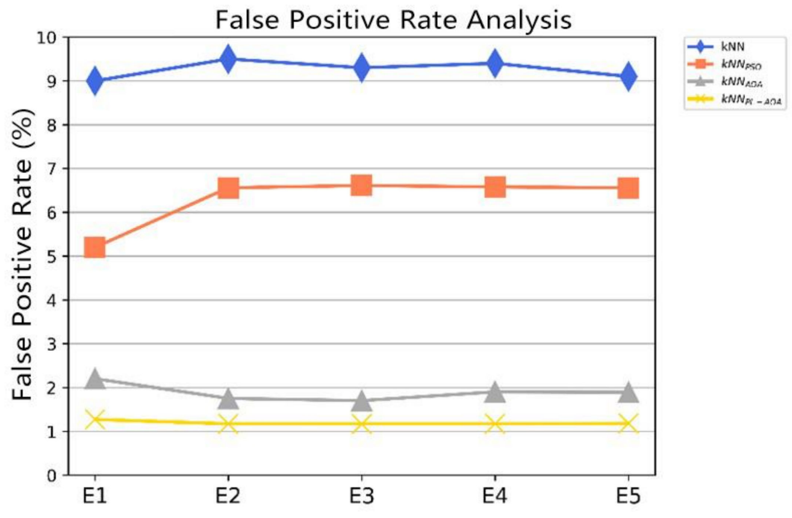

Figure 6. The false positive rate comparison for kNN, kNN$_{PSO}$, kNN$_{AOA}$, and kNN$_{PL-AOA}$.

Figure 7 shows the AOC curves of the four classification methods, where we can see that the kNN$_{PL-AOA}$ proposed in the article achieves a good effect.

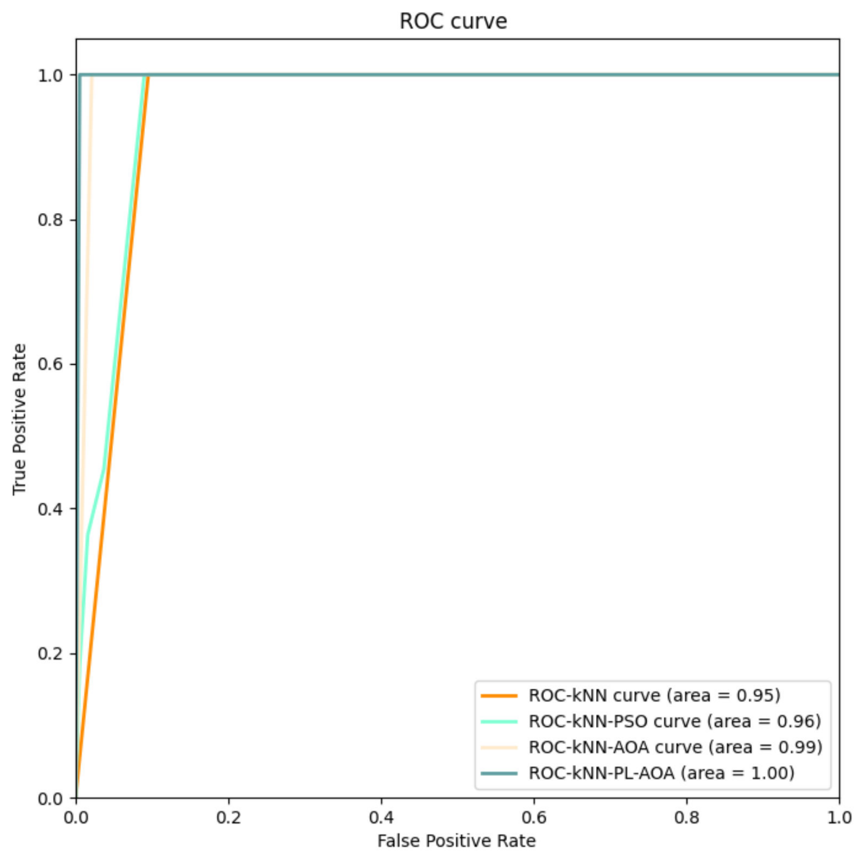

Figure 7. The AOC curves for kNN, $\mathrm{kNN}_{\mathrm{PSO}}$, $\mathrm{kNN}_{\mathrm{AOA}}$, and $\mathrm{kNN}_{\mathrm{PL-AOA}}$.

6. Conclusions

Due to the proposal of edge technology and its extensive combination with IoT devices, some complex technologies can be easily implemented in this way [49–51]. Some applications pose various security threats to WSNs, especially in unattended environments. In order to ensure the security and reliability of WSN services, an intrusion detection system (IDS) should be established. Intrusion detection is one of the key issues that urgently needs to be resolved in the practical application of WSNs. This paper proposes an edge-intelligent intrusion detection system that can be applied when a WSN encounters a DoS attack. First, we improve the AOA algorithm by using Lévy flight to improve the ability to jump out of the local optimum, and we use a parallel strategy to improve the population diversity in iterations. Then, through the combination of the improved PL-AOA optimization algorithm and kNN machine learning classifier, we not only improve the accuracy of detection and classification but also greatly improve the detection precision. The improved PL-AOA algorithm proposed in this paper has passed 12 benchmark function tests with outstanding results; we achieve the best results 9 times in 12 benchmark functions. In addition, the proposed intrusion detection model has been proven to be feasible in simulation experiments using WSN standard intrusion detection data sets to achieve a 99% ACC.

Time complexity is one of the evaluation indicators to measure the pros and cons of an algorithm. The time complexity of the PL-AOA proposed in this paper consists of two parts: initialization and solution update. The time complexity of the initialization process is $O(G \times N/G) = O(G)$. Due to the parallel strategy, regardless of how many groups are divided, the final population size is still N, G is the number of groups, and the solution update is $O(N \times T) + O(T \times N \times D)$, where T denotes the iteration and D denotes the dimension. The time complexity of Lévy flight is $O(1)$. Therefore, the time complexity of the PL-AOA algorithm is $O(N \times (TD + 1))$, as with the original AOA algorithm. The time complexity of kNN is $O(N \times D)$. In the proposed $\mathrm{kNN}_{\mathrm{PL-AOA}}$, kNN is used as the update solution, so the time complexity of kNN_{PL-AOA} is also $O(N \times (TD + 1))$. Although the time complexity is a little higher than that of the native kNN, the accuracy is improved by nearly 10%, so we believe that this time loss is an appropriate trade-off.

In the future, we will focus on developing an unsupervised or semi-supervised WSN intrusion detection model, such as k-means optimized by an evolutionary algorithm, and so on. These models will not only target a particular type of DoS attack, but also strive to cover Sybil attacks, routing attacks, and other possible attacks.

Author Contributions: Introduction, G.L. (Gaoyuan Liu), H.Z. and S.N.; Methodology, G.L. (Gaoyuan Liu) and F.F.; Software, S.N.; Validation, G.L. (Gaoyuan Liu), H.Z., F.F., G.L. (Gang Liu), and Q.X.; Formal analysis, G.L. (Gaoyuan Liu) and H.Z.; Resources, F.F.; Data curation, G.L. (Gang Liu) and F.F.; Writing—original draft preparation, G.L. (Gaoyuan Liu); Writing—review and editing, G.L. (Gaoyuan Liu) and H.Z.; Supervision, F.F., G.L. (Gang Liu), Q.X. and S.N.; Project administration, H.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Research and Industrialization of Key Security Technologies of Industrial Internet for Discrete Manufacturing, 2019JZZY010132.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data set is derived from 'WSN-DS: A dataset for intrusion detection systems in wireless sensor networks'. https://doi.org/10.1155/2016/4731953 (accessed on 3 December 2021).

Acknowledgments: Thanks to H.Z. for project funding support and Tai'an Engineering Laboratory for providing the computing equipment.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Haseeb, K.; Din, I.U.; Almogren, A.; Islam, N. An Energy Efficient and Secure IoT-Based WSN Framework: An Application to Smart Agriculture. *Sensors* **2020**, *20*, 2081. [CrossRef] [PubMed]
2. Creech, G.; Hu, J. A Semantic Approach to Host-Based Intrusion Detection Systems Using Contiguousand Discontiguous System Call Patterns. *IEEE Trans. Comput.* **2014**, *63*, 807–819. [CrossRef]
3. Vokorokos, L.; BaláŽ, A. Host-Based Intrusion Detection System. In Proceedings of the 2010 IEEE 14th International Conference on Intelligent Engineering Systems, Las Palmas, Spain, 5–7 May 2010; pp. 43–47. [CrossRef]
4. Yeung, D.Y.; Ding, Y. Host-Based Intrusion Detection Using Dynamic and Static Behavioral Models. *Pattern Recognit.* **2003**, *36*, 229–243. [CrossRef]
5. Vigna, G.; Kemmerer, R.A. NetSTAT: A Network-Based Intrusion Detection System. *J. Comput. Secur.* **1999**, *7*, 37–71. [CrossRef]
6. Bivens, A.; Palagiri, C.; Smith, R.; Szymanski, B.; Embrechts, M. Network-Based Intrusion Detection Using Neural Networks. In Proceedings of the Intelligent Engineering Systems through Artificial Neural Networks, St. Louis, MO, USA, 10–13 November 2002; ASME Press: New York, NY, USA, 2002; pp. 579–584.
7. Snapp, S.R.; Brentano, J.; Dias, G.V.; Goan, T.L.; Heberlein, L.T.; Ho, C.-L.; Levitt, K.N.; Mukherjee, B.; Smaha, S.E.; Grance, T.; et al. DIDS (Distributed Intrusion Detection System)—Motivation, Architecture, and an Early Prototype. In Proceedings of the 14th National Computer Security Conference, Washington, DC, USA, 1–4 October 1991; pp. 1–9.
8. Farooqi, A.H.; Khan, F.A. A Survey of Intrusion Detection Systems for Wireless Sensor Networks. *Int. J. Ad Hoc Ubiquitous Comput.* **2012**, *9*, 69–83. [CrossRef]
9. Doumit, S.S.; Agrawal, D.P. Self-Organized Criticality & Stochastic Learning Based Intrusion Detection System for Wireless Sensor Networks. In Proceedings of the IEEE Military Communications Conference, MILCOM 2003, Boston, MA, USA, 13–16 October 2003; pp. 609–614. [CrossRef]
10. Tylman, W. Misuse-Based Intrusion Detection Using Bayesian Networks. *Int. J. Crit. Comput. Syst.* **2010**, *1*, 178–190. [CrossRef]
11. García-Teodoro, P.; Díaz-Verdejo, J.; Maciá-Fernández, G.; Vázquez, E. Anomaly-Based Network Intrusion Detection: Techniques, Systems and Challenges. *Comput. Secur.* **2009**, *28*, 18–28. [CrossRef]
12. Aljawarneh, S.; Aldwairi, M.; Yassein, M.B. Anomaly-Based Intrusion Detection System through Feature Selection Analysis and Building Hybrid Efficient Model. *J. Comput. Sci.* **2018**, *25*, 152–160. [CrossRef]
13. Sermanet, P.; Chintala, S.; Lecun, Y. Convolutional Neural Networks Applied to House Numbers Digit Classification. In Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012), Tsukuba, Japan, 11–15 November 2012; pp. 3288–3291.
14. Breiman, L.E.O. Random Forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]

15. Lewis, D.D. Naive (Bayes) at Forty: The Independence Assumption in Information Retrieval. In Proceedings of the 10th European Conference on Machine Learning, Chemnitz, Germany, 21–23 April 1998; Springer: Berlin/Heidelberg, Germany, 1998; Volume 1398, pp. 4–15.
16. Safavian, S.R.; Landgrebe, D. A survey of decision tree classifier methodology. *IEEE Trans. Syst. Man Cybern.* **1991**, *21*, 660–674. [CrossRef]
17. Fukunaga, K.; Narendra, P.M. A Branch and Bound Algorithm for Computing K-Nearest Neighbors. *IEEE Trans. Comput.* **1975**, *100*, 750–753. [CrossRef]
18. Zhang, G.; Wang, T.; Wang, G.; Liu, A.; Jia, W. Detection of Hidden Data Attacks Combined Fog Computing and Trust Evaluation Method in Sensor-Cloud System. *Concurr. Comput. Pract. Exp.* **2021**, *33*, 1. [CrossRef]
19. Khan, M.A.; Khan, M.A.; Jan, S.U.; Ahmad, J.; Jamal, S.S.; Shah, A.A.; Pitropakis, N.; Buchanan, W.J. A Deep Learning-Based Intrusion Detection System for Mqtt Enabled Iot. *Sensors* **2021**, *21*, 7016. [CrossRef]
20. Kelli, V.; Argyriou, V.; Lagkas, T.; Fragulis, G.; Grigoriou, E.; Sarigiannidis, P. Ids for Industrial Applications: A Federated Learning Approach with Active Personalization. *Sensors* **2021**, *21*, 6743. [CrossRef]
21. Tan, S. An Effective Refinement Strategy for kNN Text Classifier. *Expert Syst. Appl.* **2006**, *30*, 290–298. [CrossRef]
22. Liang, X.; Gou, X.; Liu, Y. Fingerprint-Based Location Positoning Using Improved kNN. In Proceedings of the 2012 3rd IEEE International Conference on Network Infrastructure and Digital Content, Beijing, China, 21–23 September 2012; pp. 57–61. [CrossRef]
23. Chen, M.; Guo, J.; Wang, C.; Wu, F. PSO-based adaptively normalized weighted kNN classifier. *J. Comput. Inf. Syst.* **2015**, *11*, 1407–1415. [CrossRef]
24. Xu, H.; Fang, C.; Cao, Q.; Fu, C.; Yan, L.; Wei, S. Application of a Distance-Weighted KNN Algorithm Improved by Moth-Flame Optimization in Network Intrusion Detection. In Proceedings of the 2018 IEEE 4th International Symposium on Wireless Systems within the International Conferences on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS-SWS), Lviv, Ukraine, 20–21 September 2018; pp. 166–170. [CrossRef]
25. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey Wolf Optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]
26. Tahir, M.A.; Bouridane, A.; Kurugollu, F. Simultaneous Feature Selection and Feature Weighting Using Hybrid Tabu Search/K-Nearest Neighbor Classifier. *Pattern Recognit. Lett.* **2007**, *28*, 438–446. [CrossRef]
27. Glover, F.; Laguna, M. *Tabu Search, Handbook of Combinatorial Optimization*; Springer: Boston, MA, USA, 1998; pp. 2093–2229.
28. Whitley, D. A Genetic Algorithm Tutorial. *Stat. Comput.* **1994**, *4*, 65–85. [CrossRef]
29. Storn, R.; Price, K. Differential Evolution—A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [CrossRef]
30. Mirjalili, S.; Lewis, A. The Whale Optimization Algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [CrossRef]
31. Chu, S.-C.; Tsai, P.; Pan, J.-S. Cat Swarm Optimization. In Proceedings of the 9th Pacific Rim International Conference on Artificial Intelligence, Guilin, China, 7–11 August 2006; Springer: Berlin/Heidelberg, Germany, 2006; pp. 854–858. [CrossRef]
32. Mirjalili, S.; Mirjalili, S.M.; Hatamlou, A. Multi-Verse Optimizer: A Nature-Inspired Algorithm for Global Optimization. *Neural Comput. Appl.* **2016**, *27*, 495–513. [CrossRef]
33. Meng, Z.; Pan, J.S.; Xu, H. QUasi-Affine TRansformation Evolutionary (QUATRE) Algorithm: A Cooperative Swarm Based Algorithm for Global Optimization. *Knowl.-Based Syst.* **2016**, *109*, 104–121. [CrossRef]
34. Abualigah, L.; Diabat, A.; Mirjalili, S.; Abd Elaziz, M.; Gandomi, A.H. The Arithmetic Optimization Algorithm. *Comput. Methods Appl. Mech. Eng.* **2021**, *376*, 113609. [CrossRef]
35. Wolpert, D.H.; Macready, W.G. No Free Lunch Theorems for Optimization. *IEEE Trans. Evol. Comput.* **1997**, *1*, 67–82. [CrossRef]
36. Iliyasu, A.M.; Fatichah, C. A Quantum Hybrid PSO Combined with Fuzzy K-NN Approach to Feature Selection and Cell Classification in Cervical Cancer Detection. *Sensors* **2017**, *17*, 2935. [CrossRef]
37. Callahan, P.B.; Kosaraju, S.R. A Decomposition of Multidimensional Point Sets with Applications to K-Nearest-Neighbors and N-Body Potential Fields. *J. ACM* **1995**, *42*, 67–90. [CrossRef]
38. Rajagopalan, B.; Lall, U. A K-Nearest-Neighbor Simulator for Daily Precipitation and Other Weather Variables. *Water Resour. Res.* **1999**, *35*, 3089–3101. [CrossRef]
39. Yang, X.S.; Deb, S. Cuckoo Search via Lévy Flights. In Proceedings of the 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC), Coimbatore, India, 9–11 December 2009; pp. 210–214. [CrossRef]
40. Chang, J.F.; Chu, S.C.; Roddick, J.F.; Pan, J.S. A Parallel Particle Swarm Optimization Algorithm with Communication Strategies. *J. Inf. Sci. Eng.* **2005**, *21*, 809–818.
41. Cheng, D.; Zhang, S.; Deng, Z.; Zhu, Y.; Zong, M. κ NN Algorithm with Data-Driven k Value. In Proceedings of the International Conference on Advanced Data Mining and Applications, 10th International Conference, ADMA 2014, Guilin, China, 19–21 December 2014; Lecture Notes in Computer Science. Springer: Cham, Switzerland, 2014; Volume 8933, pp. 499–512. [CrossRef]
42. Kaur, T.; Saini, B.S.; Gupta, S. An Adaptive Fuzzy K-Nearest Neighbor Approach for MR Brain Tumor Image Classification Using Parameter Free Bat Optimization Algorithm. *Multimed. Tools Appl.* **2019**, *78*, 21853–21890. [CrossRef]
43. Pan, J.S.; Sun, X.X.; Chu, S.C.; Abraham, A.; Yan, B. Digital Watermarking with Improved SMS Applied for QR Code. *Eng. Appl. Artif. Intell.* **2021**, *97*, 104049. [CrossRef]

44. Marriwala, N.; Rathee, P. An Approach to Increase the Wireless Sensor Network Lifetime. In Proceedings of the 2012 World Congress on Information and Communication Technologies, Trivandrum, India, 30 October–2 November 2012; pp. 495–499. [CrossRef]
45. Mukherjee, B.; Heberlein, L.T.; Levitt, K.N. Network Intrusion Detection. *IEEE Netw.* **1994**, *8*, 26–41. [CrossRef]
46. Shi, Y.; Tian, Y.; Kou, G.; Peng, Y.; Li, J. Network Intrusion Detection. In *Optimization Based Data Mining: Theory and Applications*; Advanced Information and Knowledge Processing; Springer: London, UK, 2011; pp. 237–241. [CrossRef]
47. Mirjalili, S. SCA: A Sine Cosine Algorithm for Solving Optimization Problems. *Knowl.-Based Syst.* **2016**, *96*, 120–133. [CrossRef]
48. Almomani, I.; Al-Kasasbeh, B.; Al-Akhras, M. WSN-DS: A Dataset for Intrusion Detection Systems in Wireless Sensor Networks. *J. Sens.* **2016**, *2016*, 4731953. [CrossRef]
49. Otoum, S.; Kantarci, B.; Mouftah, H.T. On the Feasibility of Deep Learning in Sensor Network Intrusion Detection. *IEEE Netw. Lett.* **2019**, *1*, 68–71. [CrossRef]
50. Almaiah, M.A. *A New Scheme for Detecting Malicious Attacks in Wireless Sensor Networks Based on Blockchain Technology*; Springer: Cham, Switzerland, 2021; pp. 217–234. [CrossRef]
51. Sajjad, S.M.; Bouk, S.H.; Yousaf, M. Neighbor Node Trust Based Intrusion Detection System for WSN. *Procedia Comput. Sci.* **2015**, *63*, 183–188. [CrossRef]