Available online at www.ijpe-online.com

International Journal of Performability Engineering

vol. 15, no. 6, June 2019, pp. 1600-1610
DOI: 10.23940/ijpe.19.06.p11.16001610

Applying an Improved Elephant Herding Optimization Algorithm with Spark-based Parallelization to Feature Selection for Intrusion Detection

Hui Xu*, Qianqian Cao, Heng Fu, and Hongwei Chen

School of Computer Science, Hubei University of Technology, Wuhan, 430068, China

Abstract

With the growth of the intrusion data scale model, irrelevant or redundant features in high-dimensional intrusion detection data leads to slow processing speed of the intrusion detection algorithm, and the consumption of the algorithm in time and space will increase as the feature dimensions increase. In view of good classification performance of the Elephant Herding Optimization (EHO) algorithm in reducing feature redundancy, this paper introduces the EHO algorithm into feature selection for intrusion detection. Since the basic EHO algorithm tends to fall into a local optimum and lacks strong search ability, the classification performance and dimensional reduction ability of the algorithm are severely limited. Therefore, an Improved Elephant Herding Optimization (IEHO) algorithm is proposed in this paper to search the feature space and find the optimal feature subset, so that the feature number is minimized while the classification performance is maximized. As the scale of intrusion data grows, the large amount of redundant information in the intrusion data will cause the improved algorithm to process slowly. Thus, in this case, the improved algorithm is considered to be parallelized to relieve the pressure of single-machine operation. This paper then proposes a Spark-based distributed parallel IEHO algorithm for intrusion detection, and a feature selection method based on this algorithm for intrusion detection is discussed. The feature selection in a distributed environment can improve the running efficiency of the IEHO algorithm, so as to reduce the running time of the algorithm under the premise of ensuring classification accuracy. As for the experimental validation, both UCI and KDD CUP99 datasets are used to verify the feature selection for intrusion detection. Compared with the classical PSO, MFO, and EHO algorithms, the feature selection by the binary IEHO algorithm is improved by 4.16%, 1.42%, and 0.98%, respectively, and the classification performance is also significantly improved. Compared with the stand-alone version of the IEHO algorithm, the classification efficiency of the parallel IEHO algorithm based on Spark for intrusion feature selection is significantly improved, and the acceleration ratio is increased by two orders of magnitude.

Keywords: network intrusion detection; feature selection; Elephant Herding Optimization algorithm; Spark-based parallelization

(Submitted on March 20, 2019; Revised on April 7, 2019; Accepted on June 5, 2019)

1. Introduction

Intrusion detection technology discovers whether there are any violations of security policies in the network or system by analyzing network traffic, system audit records, and data packets in the network [1-2]. In intrusion detection, the network data is large and contains a large number of irrelevant and redundant features; these features do not contain or contain little state information, regardless of detection. They can reduce the accuracy of classification or clustering as well as increase the time and space complexity of learning and training, which affects the operating efficiency of the algorithm. Research shows that feature selection can remove redundant features on the basis of maintaining the integrity of original network data. Some scholars have reduced the feature dimensions of intrusion data through feature selection to improve the accuracy of network intrusion detection, and some achievements have been made [3-4]. Reference [5] proposed an ant-lion algorithm for feature selection of intrusion detection. Reference [6] used the particle swarm optimization algorithm to obtain feature subsets for intrusion detection; compared with the original feature set, these feature subsets can effectively improve the accuracy of intrusion detection. However, due to the complex network data structure and large amount of data, the feature dimension is high. It is an NP-hard combinatorial optimization problem to select an effective feature subset from the network original data feature set. These optimization algorithms will fall into local optimization or premature convergence in the iterative process, which limits the optimization ability of the algorithm. It is necessary to further improve the classification

* Corresponding author.
E-mail address: xuhui@mail.hbut.edu.cn

performance of the algorithm to obtain an effective feature subset and improve the classification accuracy of intrusion detection feature selection.

Elephant Herding Optimization (EHO) is a new meta-heuristic search algorithm based on elephant herding behavior, and it is used to solve the global unconstrained optimization problem [7]. Although the EHO algorithm is a relatively new meta-heuristic algorithm, it has the characteristics of simple structure, few control parameters, and easy combination with other methods. It has been successfully applied to multi-level thresholds, support vector machine parameter optimization, and many other problems [8-9]. However, the EHO algorithm has poor global search ability, and it is easy for it to fall into a local optimum. When dealing with high-dimensional data samples, the classification performance and operational efficiency of the EHO algorithm are limited. In this paper, an improved Elephant Herding Optimization (IEHO) algorithm is proposed for intrusion detection feature selection. The spatial position of the elephant group is initialized by Levy's flight, and the EHO algorithm avoids falling into a local optimum by introducing Levy's flight strategy. The PSO algorithm has great advantages in processing high-dimensional data. Therefore, the particle swarm search strategy is introduced to improve the global search ability of the EHO algorithm [10]. The IEHO algorithm is used to select the feature attributes in the original network packets, and the global search ability of IEHO algorithm is used to conduct a comprehensive search of the feature space. The irrelevant and redundant feature attributes are removed to achieve better classification accuracy and select the optimal feature subset for intrusion detection. Experimental results show that compared with other feature selection algorithms, the IEHO algorithm has better classification performance and dimension reduction ability.

Many studies have shown that it is feasible to combine intrusion detection technology with traditional machine learning methods to detect intrusion behavior. Using machine learning algorithms such as neural network algorithms and intelligent optimization algorithms such as PSO to process host-based and network-based data, the accuracy and scalability of intrusion detection can be effectively improved, but these algorithms cannot solve the timely processing of massive data [11]. With the continuous development of distributed heterogeneity of computer networks, network intrusion behavior also tends to be diversified and distributed gradually. As the scale of the network continues to expand and the scale of data grows exponentially, the high performance of parallel processing of data in a distributed environment can improve the efficiency of intrusion detection [12-13]. Reference [14] proposed a cloud computing-based intrusion detection model. Reference [15] proposed a cloud computing-based intrusion detection mechanism for the problems faced by traditional intrusion detection systems, and experiments showed that this method can effectively improve the detection speed. Intrusion detection systems in stand-alone environments have been unable to effectively handle large-scale intrusion data, and the intrusion detection system must be distributed to meet real-time requirements. Spark is an iterative computing distributed framework implemented by Scala language that can realize machine learning, graph computing, and other functions. Due to its characteristic of having elastic distributed datasets of RDD, it saves a large amount of disk I/O input and output operations compared with MapReduce, and it also improves the running speed [16-17]. Based on the Spark big data framework, this paper parallelizes the IEHO algorithm to improve the running efficiency of the algorithm and proposes an intrusion detection feature selection method of the IEHO algorithm based on Spark parallelization.

2. Application of IEHO Algorithm in Feature Selection of Intrusion Detection

2.1. The Elephant Herding Optimization Algorithm (*EHO*)

The Elephant Herding Optimization (EHO) algorithm is a new swarm intelligence optimization algorithm proposed by Wang et al. in 2016. It was derived from the animal husbandry behavior of elephants in nature [18]. In the EHO algorithm, the update operation is first performed to determine the search direction and the local search detail level of the algorithm, and then the separation operation is implemented. In view of the limitations of the paper, it will not be described in detail here.

Randomly initialize the elephant population, divide the elephant population into n clans, and each clan has j elephant individuals. In each iteration, the position of each elephant j is defined as Equation (1).

$$x_{new,ci,j} = x_{ci,j} + \alpha \cdot (x_{best,ci} - x_{ci,j}) \cdot r \tag{1}$$

The position of the female matriarch $x_{best,ci}$ is defined as Equation (2).

$$x_{new,ci,j} = \beta \times x_{center,ci} \tag{2}$$

The center of the elephant clan is defined as Equation (3).

$$x_{center,ci,d} = \frac{1}{n_{ci}} \cdot \sum_{j=1}^{n_{ci}} x_{ci,j,d} \tag{3}$$

The elephant position with the worst fitness value is defined as Equation (4).

$$x_{worst,ci} = x_{\min} + (x_{\max} - x_{\min} + 1) \times rand \tag{4}$$

2.2. The Improved Elephant Herding Optimization Algorithm (*IEHO*)

2.2.1. Introducing Levy Flight Strategy in the Update Mechanism

Levy Flight [19] is a non-Gaussian randomization process that uses the frequent short-range search in Levy flight to search carefully around the current optimal solution and improve the local search ability. Expand the search range by using an occasional long-distance jump search to enhance the global search capabilities and avoid falling into local optimal solutions. The Levy flight path expression derived using the Mantegna algorithm is defined as Equation (5).

$$Levy(\lambda) = \frac{r_1(1,d) \times \sigma}{\left| r_2(1,d) \right|^{1/\beta}} \tag{5}$$

In the EHO algorithm, the location update of the individual in the algorithm will be affected by the location of the elephant with the best fitness. If the current optimal individual is attracted by the local extremum, the algorithm has no effective mechanism to get rid of the constraint, which will lead to the algorithm being attracted to the local extreme value prematurely and falling into the local optimum. Inspired by Levy's flight strategy, this paper uses the levy flight behavior model to simulate the elephant's position update behavior, making full use of the levy flight to expand the search range to avoid the elephant individual being attracted by a local extremum. This makes the elephant individuals move in a better direction and allow the algorithm to avoid premature convergence and falling into local optimal. The improved location update operation is described as Equation (6).

$$x_{new,ci,j} = x_{ci,j} + Levy(\lambda) + \alpha \cdot (x_{best,ci} - x_{ci,j}) \cdot r \tag{6}$$

2.2.2. Introducing PSO Strategy in the Separation Mechanism

PSO is a population-based meta-heuristic intelligent optimization algorithm that was proposed by Kennedy et al. in 1995 based on the foraging behavior of birds [20]. In view of the limitations of the paper, it will not be described in detail here.

In the EHO algorithm, leaving male elephants can improve the diversity of the population. The EHO algorithm uses Equation (4) to replace the individual with the worst fitness value in the population. The results are too random, which reduces the global search ability and the diversity of the population to some extent. Due to the simple parameter adjustment, fast convergence, and wide range of global search capabilities of the PSO algorithm, it can be embedded in the separation operation of the EHO algorithm. The specific updating strategy is as follows: at each iteration, the male elephant with the worst fitness value in the population is regarded as each particle in the particle swarm. The particle swarm optimization strategy is used to optimize the search space by using the inertia of the particles. Calculate the fitness function value of each particle. If a better fitness value is obtained, update $x_{worst,ci}$; otherwise, it will not be updated. The location update equation is shown in Equations (7) and (8).

$$v_i^{t+1} = \omega \cdot v_i^t + c_1 \cdot r_1 (x_{best,ci}^t - x_{worst,ci}^t) + c_2 \cdot r_2 (x_{g,ci}^t - x_{worst,ci}^t) \tag{7}$$

$$x_i^{t+1} = x_{worst,ci}^t + v_i^{t+1} \tag{8}$$

2.3. Feature Selection

Feature selection is one of the important data processing steps for intrusion detection. Feature selection can be summarized as follows: select a set of subsets from the original dataset, so that the classification effect of this set of subsets is as good as possible or the corresponding fitness function value is as high as possible [21-22]. The mathematical model can be defined as follows: for a given set of data samples $F=\{f_1, f_2, \cdots, f_n\}$, n is the size of the feature set, and any one of its feature subsets can be represented by a binary vector $S=\{s_1, s_2, \cdots, s_n\}$, $s_i \in \{0,1\}$, $i=1,2,\cdots,n$, and $s_i=1$ indicates that the i^{th} feature is selected by f_i; otherwise, $s_i=0$ indicates that the i^{th} feature is not selected by f_i.

The ultimate goal of feature selection is to obtain higher classification results with the least number of features. The two key issues that need to be addressed are designing search strategies and evaluation functions for generating and evaluating feature subsets. The fitness function, also known as the evaluation function, is used to judge how good or bad the solution is represented by each individual in the population. In intrusion detection, the fitness function is mainly related to the number of features selected and the accuracy of classification. Therefore, the fitness function of feature selection is defined as Equation (9).

$$F_i = \frac{accuracy(i)}{1+\eta \cdot n(i)} \tag{9}$$

Where $F(i)$ is the fitness value of the feasible solution, $accuracy(i)$ represents the classification accuracy, $n(i)$ represents the selected number of features, and η is a weight parameter, which is usually set as $\eta = 0.01$.

2.4. Intrusion Detection Feature Selection Method based on Binary IEHO Algorithm

The IEHO improves the convergence speed and global optimization ability of the EHO algorithm, thereby improving the classification performance of the algorithm. Since the feature selection is a binary combination optimization problem and the IEHO algorithm is still used to handle continuous problems, it cannot be directly used for feature selection. Therefore, this paper combines binary coding and the IEHO algorithm and proposes an improved intrusion detection feature selection method based on the binary IEHO algorithm.

In the iterative process of binary IEHO algorithm feature selection, the elephant group searches the solution space, and an elephant individual is a search agent. The binary IEHO algorithm adopts a binary coding form, and each elephant's position corresponds to a feasible solution. Each dimension of the elephant is binary coded, and each dimension of the code takes a value of 0 or 1, where "0" represents that the feasible solution represented by the elephant does not select the feature and "1" represents this feature on behalf of the feasible solution represented by the elephant individual. The binary vector is constructed using the sigmoid function, which is shown below,

$$sig(x_{new,ci,j}) = \frac{1}{1+e^{-(x_{new,ci,j})}} \tag{10}$$

Therefore, the binary position of the elephant in Equation (7) can be expressed as

$$x_{new,ci,j} = \begin{cases} 1, & \text{if } sig(x_{new,ci,j}) > rand() \\ 0, & \text{otherwise} \end{cases} \tag{11}$$

The values of elephant group individuals can be mapped to [0,1] through the transformation of Equations (10) and (11), so as to represent the probability of taking 0 or 1 after feature selection using the binary IEHO algorithm. The updated binary values of elephant individuals in each dimension are obtained.

3. The Improved IEHO Algorithm based on Spark Parallelization

3.1. Introduction to Spark

Apache Spark is a big data parallel computing framework based on in-memory computing. It is good at handling iterative

machine learning and interactive query. It can effectively improve the efficiency of processing massive data and guarantee high fault tolerance and scalability. Its main advantage is that it can use multiple machine-based memory-based parallel iterative execution operations to achieve fast processing of big data [23]. Spark is an open source distributed computing framework based on the MapReduce algorithm. It is fully compatible with Hadoop's original ecosystem and uses HDFS as the distributed storage system. However, unlike Hadoop, Spark is mainly based on memory computing and is iterative. Spark can save the intermediate results generated during the running process in memory without having to read HDFS files, which saves disk I/O time. However, the MapReduce computing model requires multiple accesses to the disk during iterative processing, which affects the training speed. Therefore, the computing speed based on Spark memory is faster than Hadoop, so Spark is better at processing algorithms that need to be iterated in data mining and machine learning [24]. Resilient distributed dataset (RDD) is the core of Spark's data storage. It can efficiently share data in iterative computing in order to provide an abstraction for working set-based applications [25].

3.2. Implementation of IEHO Algorithm based on Spark Parallelization

When dealing with large-scale and high-dimensional intrusion detection data, the IEHO algorithm in the stand-alone environment requires a large number of iterations to converge to better results. When performing intrusion data feature selection, the classification effect is not obvious and the operation efficiency is very low. Intrusion detection systems in stand-alone environments can no longer effectively handle large-scale intrusion data, while the high performance of parallel processing data in a distributed environment can improve the efficiency of intrusion detection. Spark is a distributed computing framework based on memory computing that can efficiently process big data. In order to improve the classification efficiency of the IEHO algorithm for feature selection in intrusion detection, this paper uses the computing interface provided by RDD of the Spark framework to realize the parallelization of the IEHO algorithm.

In the iterative process of the IEHO algorithm based on Spark distributed parallelization, we parallelize the process of elephant iterative search for the optimal solution. The position of each elephant and the process of finding the optimal solution are called an independent parallel unit, N elephants form N independent parallel units, which are processed in parallel using Spark. Since the operation of Spark is based on the RDD dataset, to achieve parallelization of the IEHO algorithm on Spark, the dataset needs to be transferred to the RDD, and then the feature selection is based on the RDD. The IEHO algorithm based on Spark parallelization is named SPIEHO. The algorithm SPIEHO algorithm flow chart is shown in Figure 1.

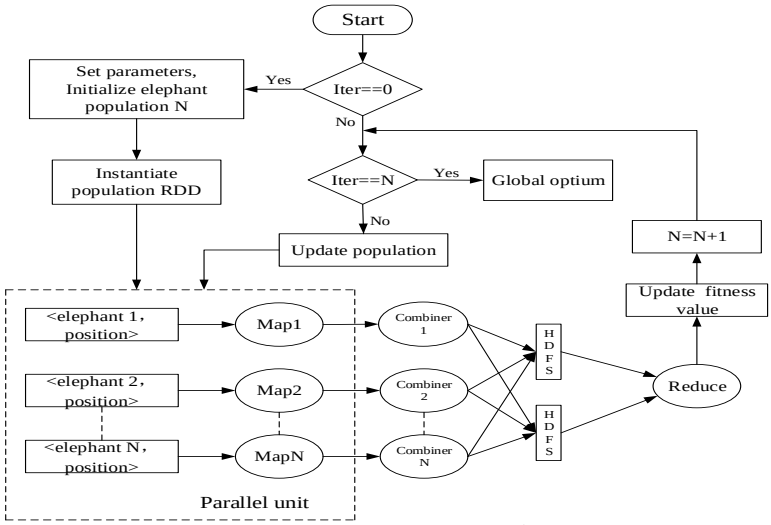


Figure 1. Spark-based SPIEHO algorithm flow chart

3.3. MapReduce Pseudo-Code for Feature Selection of Parallel SPIEHO Algorithm

Algorithms 1 and 2 are the MapReduce pseudo-codes realized by the SPIEHO algorithm on Spark parallelization.

Algorithm 1 is the Map iteration process of the SPIEHO algorithm. In Spark, each Map function sequentially reads the dataset of each partition stored in the HDFS from local input in the form of (key, value). In the parallelized SPIEHO algorithm, the Map function corresponds to < elephant, position > and runs iteratively in parallel mode to calculate the fitness value of elephants in different partitions, and the obtained fitness value is summed in the Combiner with N elephants.

Algorithm 1: SPIEHO-Map

Input: key<input file offset, $x_{ci,j}$ >; value<partition instances, fitness of each elephant>

Output: key<partition pos, $x_{ci,j}$ >; value<new fitness value of each elephant for its partition>;

1. **For** all clans ci do
2. **For** all solution *j* in the clan *ci* do
3. /*update $x_{ci,j}$ and generate $x_{new,ci,j}$ */
4. $x_{new,ci,j} \leftarrow x_{ci,j} + Levy(\lambda) + \alpha \cdot (x_{best,ci} - x_{ci,j}) \cdot r$
5. /*update $x_{best,ci}$ and generate $x_{new,ci}$ */
6. $x_{new,ci,j} \leftarrow \beta \times x_{center,ci}$;
7. /*evaluate current fitness value*/
8. $f_{best} \leftarrow f_{best}$ /(1+ *Att*. /100); /* *Att*. :instance Test.numAttributes*/
9. $key \leftarrow (partition\ id, x_{new,ci,j})$;
10. value $\leftarrow f_{best}$;
11. Emit<key, value>;
12. **End For**
13. **End For**

Algorithm 2 is the Reduce pseudocode of the SPIEHO algorithm. The Reduced phase of SPIEHO algorithm obtains the value stored in the Combiner in the form of (composite key, values). That is, the Reduce function merges all the calculated values corresponding to the same key and calculates the corresponding fitness value for each elephant.

Algorithm 2: SPIEHO- Reduce

Input: key<partition pos, $x_{ci,j}$ >; value< f_{best} >;

Output: key<partition pos, $x_{ci,j}$ >; value< $f(f_{best})$ >;

1. **For** all clans *ci* do
2. update $x_{new,ci}$
3. **For** all solution *j* in the clan *ci* do
4. /*update the worst solution in the clan *ci**/
5. if $f(x_{new,worst,ci}) > f(x_{worst,ci})$;
6. $x_i^{t+1} = x_{ci,j}^t + v_i^{t+1}$;
7. else
8. $x_{worst,ci} = x_{\min} + (x_{\max} - x_{\min} + 1) \times rand$;
9. **End For**
10. /*rank the elephants and find the current f_{best} */
11. key $\leftarrow (partition\ pos, x_{new,ci,j})$;
12. value $\leftarrow f(f_{best})$;
13. emit<key,value>;
14. **End For**

4. Simulation Experiments and Analysis

In this paper, several algorithms are compared through two-part experiments. Firstly, the binary IEHO algorithm is used to evaluate the feature selection of intrusion detection. The second part compares the running time of the intrusion detection feature selection based on the stand-alone IEHO algorithm and the SPIEHO algorithm in the distributed environment, and it verifies the efficiency of the SPIEHO algorithm based on Spark parallelization. The attribute information of the nodes in the experimental environment is shown in Table 1.

4.1. Evaluation of Feature Selection Method based on Stand-Alone Binary IEHO Algorithm

4.1.1. Comparative Analysis of UCI Dataset Experiments

In order to verify the validity of IEHO algorithm, this paper is based on JAVA language combined with Weka, using the UCI dataset and KDD CUP99 intrusion detection dataset [26] as the experimental dataset. Firstly, the IEHO algorithm is

compared with the EHO, PSO, and MFO algorithms in the UCI dataset to verify the effectiveness. The details of the UCI datasets are shown in Table 2.

Table 1. Node information table

Attribute name	Attribute value
Number of nodes	5
Hadoop version	2.7.4
Spark version	2.0.1
Spark mode	Stand Alone
JDK version	1.8.0
Node CPU	Intel (R) Core(TM) i5-7500 CPU 3.40GHz
Node memory	16G

Table 2. UCI datasets

Dataset name	Number of features	Number of samples	Category
Wine	13	178	3
Glass	10	214	6
Ionosphere	34	351	2
Heart	13	270	2

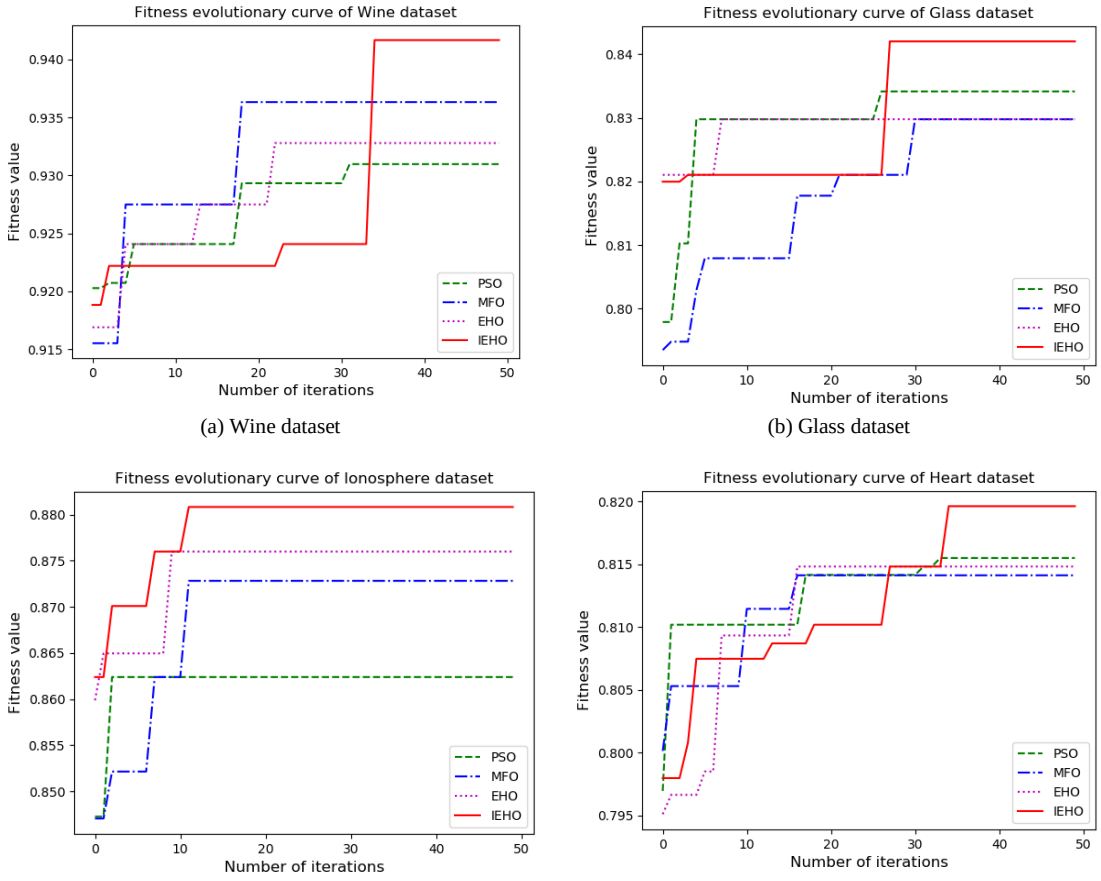
In the experiment, the four algorithms are used to select the features of the above datasets. The parameters of each algorithm are set as follows: the population size is 30 and the number of iterations is 50. The four algorithms are independently iterated for 20 times. The performance of the algorithm is shown in Table 3. The corresponding fitness value convergence is shown in Figure 2.

Table 3. Feature selection results and fitness values corresponding to different datasets

Dataset	Algorithm	Number of original features	Number of optimal feature subsets	Fitness value
Wine	PSO	13	6	0.9309
	MFO	13	6	0.9363
	EHO	13	5	0.9328
	IEHO	13	4	0.9417
Glass	PSO	10	6	0.8341
	MFO	10	6	0.8298
	EHO	10	6	0.8298
	IEHO	10	6	0.8420
Ionosphere	PSO	34	10	0.8624
	MFO	34	9	0.8728
	EHO	34	7	0.8760
	IEHO	34	5	0.8808
Heart	PSO	13	7	0.8154
	MFO	13	5	0.8141
	EHO	13	5	0.8148
	IEHO	13	4	0.8196

It can be seen from Table 3 that, on different datasets, compared with the other algorithms, the IEHO algorithm selects the least number of optimal feature subsets while ensuring classification performance. For the datasets Wine and Glass, the EHO algorithm and IEHO algorithm select the same number of feature subsets, but the fitness value of the IEHO algorithm is 0.54% and 0.79% higher than the EHO algorithm, respectively. For the dataset Heart, the fitness value of the IEHO algorithm is increased by 1.41%, 1.48%, and 1.15%, respectively, compared with the other algorithms.

It can be seen from Figure 2 that for different datasets, the fitness value obtained by the IEHO algorithm is significantly better than the other comparison algorithms, and it can avoid the defect that the original algorithm is prone to premature convergence and falling into a local optimum, which proves the effectiveness of the IEHO algorithm proposed in this paper.



(a) Wine dataset (b) Glass dataset

(c) Ionosphere dataset (d) Heart dataset

Figure 2. Fitness value convergence curve of different datasets

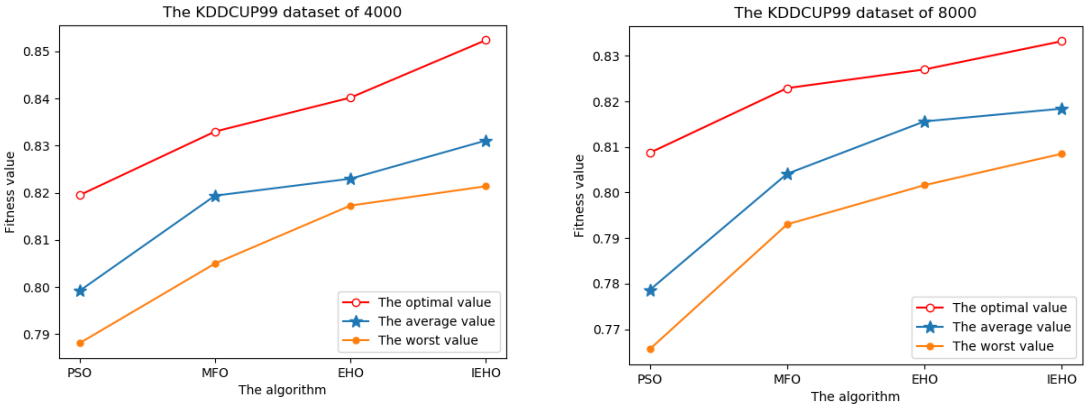
4.1.2. Comparative Analysis of KDD CUP99 Intrusion Detection Dataset Experiment

KDD CUP99 is a commonly used dataset in the field of intrusion detection. In the experiment, the KDD CUP99 datasets of size 4000 and 8000 are randomly taken, with 50% of the training set and 50% of the test set for comparison experiments. The performance of feature selection of the binary IEHO algorithm in intrusion detection is verified. The parameter settings of each algorithm in the experiment are unchanged. The experimental results of intrusion detection feature selection based on the PSO, MFO, EHO, and IEHO algorithms are shown in Table 4. The best fitness value, average fitness value, and worst fitness value contrast are shown in Figure 3.

It can be seen from Table 4 and Figure 3 that the IEHO algorithm selects the least number of optimal feature subsets compared to the other algorithms. The fitness value of the IEHO algorithm is increased by 4.16%, 1.42%, and 0.98%, respectively, compared with the PSO, MFO, and EHO algorithms. The optimal fitness value, worst fitness value, and average fitness value of the IEHO algorithm feature selection are clearly better than those of the comparison algorithms. The validity of the proposed IEHO algorithm in intrusion detection feature selection is thus verified.

Table 4. Results of feature selection by different algorithms on the KDD CUP99 dataset

Dataset	Algorithm	Number of original features	Number of optimal feature subsets	Fitness value
4000	PSO	40	12	0.8195
	MFO	40	8	0.8330
	EHO	40	8	0.8402
	IEHO	40	7	0.8524
8000	PSO	40	12	0.8087
	MFO	40	9	0.8229
	EHO	40	9	0.8270
	IEHO	40	8	0.8332



(a) The KDDCUP99 dataset of 4000 (b) The KDDCUP99 dataset of 8000

Figure 3. Fitness values of different algorithms on different sizes of KDDCUP99 dataset

4.2. Comparative Analysis of IEHO Algorithm based on Spark Parallelization

In order to prove the classification efficiency of feature selection of parallel IEHO algorithm under a distributed platform, this section designs comparative experiments and performance test experiments from different dataset sizes and different node numbers. In the experiment, the experimental cluster consists of five computing nodes, and the KDD CUP99 dataset is used as the original data source to expand and obtain different datasets, which are 100W, 200W, 500W, and 1000W. Under the same data volume, the stand-alone version and the distributed IEHO algorithm are run 15 times independently, and the average running time of the stand-alone version IEHO and SPIEHO under different data is shown in Figure 4.

Figure 4 shows that with an increase in the KDD CUP99 dataset, the iterative optimization time of the SPIEHO algorithm based on Spark parallel is gradually reduced. For small-scale datasets (that is, 100W and 200W), the degree of improvement of the algorithm is relatively small, mainly because there are too many dataset partitions, so the time spent on data partitioning and result combination will increase. When the dataset was 500W and 1000W, the improvement ratio of the algorithm reached 383.85% and 484.64% respectively, which indicated that the larger the dataset, the more obvious the parallel SPIEHO algorithm speed was improved.

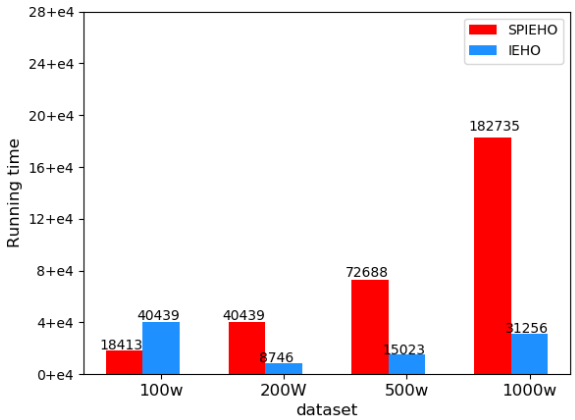


Figure 4. Comparison of IEHO and SPIEHO runtimes for different datasets

In order to better test the performance of the SPIEHO algorithm, this section also designed the average running speed and SpeedUp experiment to analyze the effect of the algorithm running in the cluster. The speedup refers to increasing the number of compute nodes in the case of processing datasets of the same size to test the performance of the parallel algorithm. In the measurement of the parallel performance of parallel algorithms, the Speedup is an important measure that can fully reflect the parallelization performance of parallel algorithms. The average running speed and SpeedUp of SPIEHO based on Spark distributed under different nodes are shown in Figures 5 and 6, respectively.

Figure 5 shows that the running time of SPIEHO algorithm will gradually decrease as the number of nodes increases.

This is because when the number of nodes increases, the nodes can process local data in parallel; with an increase in the number of nodes, the memory will also increase and more data can be stored in memory, which speeds up the operation efficiency of the algorithm.

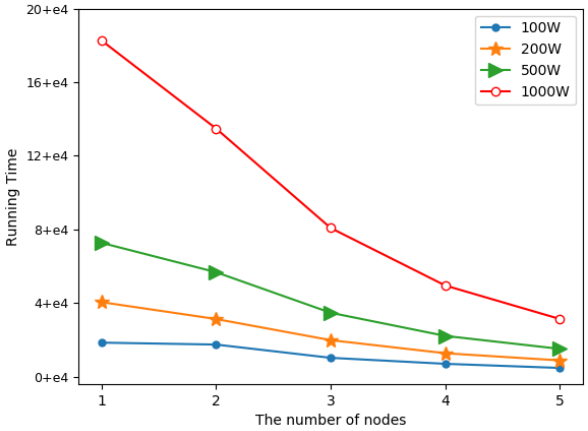


Figure 5. Average running speed of SPIEHO under different nodes

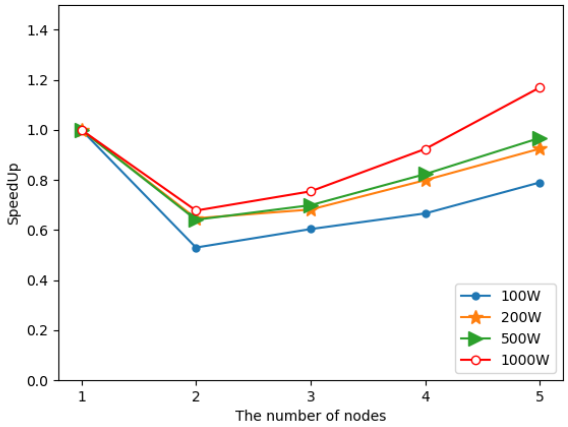


Figure 6. SpeedUp of SPIEHO under different nodes

It can be seen from the running line graph of the SpeedUp in Figure 6 that, when processing large-scale datasets, the SpeedUp of the parallel SPIEHO algorithm will increase linearly as the number of nodes increases. When the data size is relatively small (that is, 100W and 200W), the SpeedUp of the SPIEHO algorithm tends to be flat, because some nodes are in an idle state as the number of computing nodes increases. When the data size is relatively large (that is, 500W and 1000W), as the number of computing nodes increases, the SpeedUp of the algorithm will increase rapidly. This is mainly because as the number of nodes increases, more nodes can share the iterative optimization task, the memory in the cluster will gradually increase, and the RDD dataset can be stored in the memory, thereby quickly improving the operating efficiency of the algorithm.

5. Conclusions

In this paper, a feature selection method of the Elephant Herding Optimization algorithm based on Spark parallelization (SPIEHO) for intrusion detection was proposed. In view of the shortcomings of the general Elephant Herding Optimization (EHO) algorithm, an Improved Elephant Herding Optimization (IEHO) algorithm was proposed, and it was applied to the feature selection of intrusion detection to maximize the classification performance while minimizing the number of features in order to improve the classification accuracy of intrusion detection. Considering that the intrusion detection system in a stand-alone environment cannot effectively process massive intrusion detection data, the IEHO algorithm was parallelized on a Spark distributed platform to improve the speed of algorithm iterative optimization, and SPIEHO was applied to the feature selection of intrusion detection. Two-part experiments were carried out to verify that the IEHO algorithm can effectively avoid the EHO algorithm falling into a local optimum, and the IEHO algorithm can effectively improve the classification accuracy when applied to the feature selection of intrusion detection. It was proven that the SPIEHO algorithm can effectively improve the efficiency of iterative optimization while ensuring classification performance. Applying SPIEHO to feature selection of intrusion detection can effectively reduce classification time and improve detection efficiency.

Acknowledgments

This work was partly financially supported through grants from the National Natural Science Foundation of China (No. 61602162 and 61440024). The authors would like to thank all project partners for their valuable contributions and feedback.

References

1. V. Paxson and R. Sommer, “Outside the Closed World: On using Machine Learning for Network Intrusion Detection,” in *Proceedings of the 2010 IEEE Symposium on Security and Privacy*, pp. 305-316, 2010
2. B. Sun, Y. Zhang, and Z. Shang, “The Status and Trend of Intrusion Detection System Research,” in *Proceedings of the 2012 Second International Conference on Electric Information and Control Engineering*, Vol. 1, pp. 1559-1561, 2012
3. M. Dash and H. Liu, “Feature Selection for Classification,” *Intelligent Data Analysis*, Vol. 1, No. 3, pp. 131-156, 1997

4. S. W. Lin, K. C. Ying, C. Y. Lee, and Z. J. Lee, "An Intelligent Algorithm with Feature Selection and Decision Rules Applied to Anomaly Intrusion Detection," *Applied Soft Computing*, Vol. 12, No. 10, pp. 3285-3290, 2012
5. H. H. Gao, X. Y. Wang, and H. H. Yang, "Ant Colony Optimization based Network Intrusion Feature Selection and Detection," in *Proceedings of International Conference on Machine Learning & Cybernetics*, 2005
6. H. Q. Huang and H. Sun, "Intrusion Detection of Particle Swarm Selection Feature and Information Gain Determination Feature Weight," *Computer Applications*, Vol. 34, No. 6, pp. 1686-1688, 2014
7. L. D. S. Coelho, S. Deb, and X. Z. Gao, "A New Metaheuristic Optimisation Algorithm Motivated by Elephant Herding Behaviour," *International Journal of Bio-Inspired Computation*, Vol. 8, No. 6, pp. 394-409, 2017
8. A. Alihodzic, E. Tuba, and M. Tuba, "Multilevel Image Thresholding using Elephant Herding Optimization Algorithm," in *Proceedings of International Conference on Engineering of Modern Electric Systems*, pp. 240-243, 2017
9. E. Tuba and Z. Stanimirovic, "Elephant Herding Optimization Algorithm for Support Vector Machine Parameters Tuning," in *Proceedings of International Conference on Electronics, Computers and Artificial Intelligence*, pp. 1-4, 2017
10. R. K. Mallick and N. Nahak, "Hybrid Differential Evolution Particle Swarm Optimization (DE-PSO) Algorithm for Optimization of Unified Power Flow Controller Parameters," in *Proceedings of IEEE Uttar Pradesh Section International Conference on Electrical*, 2017
11. W. Zhu and Q. Zhang, "Application of Machine Learning in Network Intrusion Detection," *Data Acquisition and Processing*, Vol. 32, No. 3, pp. 479-488, 2017
12. M. X. Hua and F. J. Zhang, "Intrusion Detection System Framework in Big Data Environment," *Communications Technology*, Vol. 48, No. 11, pp. 1300-1304, 2015
13. W. Wang and J. Zhang, "Research and Implementation of Network Intrusion Detection Algorithm based on Cloud Computing Platform," *Modern Electronic Technology*, Vol. 39, No. 19, pp. 76-79, 2016
14. H. Li and Q. Wu, "A Distributed Intrusion Detection Model based on Cloud Theory," in *Proceedings of International Conference on Cloud Computing and Intelligent Systems*, pp. 435-439, 2012
15. T. L. Huang, X. Y. Liu, and X. Wang, "Research on the Intrusion Detection Mechanism based on Cloud Computing," in *Proceedings of International Conference on Intelligent Computing & Integrated Systems*, pp. 125-128, 2010
16. C. Barba-Gonz ález, J. Garcia-Nieto, A. J. Nebro, and J. F. Aldana-Montes, "Multi-Objective Big Data Optimization with Metal and Spark," in *Proceedings of International Conference on Evolutionary Multi-Criterion Optimization*, pp. 16-30, 2017
17. Z. M. Fang, Z. Y. Ruan, and P. P. Zhou, "Doppio: I/O-Aware Performance Analysis, Modeling and Optimization for In-Memory Computing Framework," in *Proceedings of IEEE International Symposium on Performance Analysis of Systems & Software*, Vol. 1, pp. 22-32, 2018
18. G. G. Wang, L. D. S. Coelho, and S. Deb, "Elephant Herding Optimization," in *Proceedings of International Symposium on Computational and Business Intelligence*, pp. 1-5, 2016
19. V. Das, S. N. Omkar, J. Senthilnath, and V. Mani, "Clustering using Levy Flight Cuckoo Search," *Advances in Intelligent Systems and Computing*, Vol. 202, pp. 65-75, 2013
20. R. Eberhart and J. Kennedy, "Particle Swarm Optimization," in *Proceedings of Icnn'95-International Conference on Neural Networks*, Vol. 4, pp. 1942-1948, 2002
21. J. Q. Wang, L. X. Zhang, Y. N. Zhao, et al., "Feature Selection in Machine Learning," *Computer Science*, Vol. 31, No. 11, pp. 180-184, 2004
22. Y. Chen, Y. Li, H. W. Shen, X. Q. Chang, "An Efficient Feature Selection Algorithm for Lightweight Intrusion Detection Systems," *Chinese Journal of Computers*, Vol. 30, No. 8, pp. 1398-1408, 2007
23. M. Chowdhury, M. J. Franklin, M. Zaharia, S. Shenker, and I. Stoica, "Spark: Cluster Computing with Working Sets," in *Proceedings of Usenix Conference on Hot Topics in Cloud Computing*, pp. 10, 2010
24. S. Q. Ruan and X. D. Wu, "Comparison of MapReduce and Spark for Big Data Analysis," *Journal of Software*, No. 6, 2018
25. J. Rini and K. K. Sherly, "Parallel Frequent Itemset Mining with Spark RDD Framework for Disease Prediction," in *Proceedings of International Conference on Circuit*, 2016
26. J. S. Wu and W. P. Zhang, "Data Analysis and Study on KDDCUP99 Dataset," *Computer Applications and Software*, Vol. 31, No. 11, pp. 321-325, 2014

Hui Xu is an associate professor in the School of Computer Science at Hubei University of Technology, Wuhan, China. Currently, her major research interests include network and service management.

Qianqian Cao is a master's student in the School of Computer Science at Hubei University of Technology. Currently, her major research interests include network security, big data, and machine learning.

Heng Fu is a master's student in the School of Computer Science at Hubei University of Technology. Currently, his major research interests include big data, data mining, and machine learning.

Hongwei Chen is a professor in the School of Computer Science at Hubei University of Technology. Currently, his major research interests include computer networks, big data, and machine learning.