Hindawi
Security and Communication Networks
Volume 2019, Article ID 1283495, 9 pages
https://doi.org/10.1155/2019/1283495

Research Article

A Neighbor Prototype Selection Method Based on CCHPSO for Intrusion Detection

Yanping Shen,[1,2] **Kangfeng Zheng**,[1] **Chunhua Wu**,[1] **and Yixian Yang**[1]

[1]*School of Cyberspace Security, Beijing University of Posts and Telecommunications, Beijing 100876, China*
[2]*School of Information Engineering, Institute of Disaster Prevention, Langfang 101601, China*

Correspondence should be addressed to Yanping Shen; shenyanping@cidp.edu.cn

Received 12 March 2019; Accepted 4 August 2019; Published 20 August 2019

Guest Editor: Fagen Li

Nearest neighbor (NN) models play an important role in the intrusion detection system (IDS). However, with the advent of the era of big data, the NN model has the disadvantages of low efficiency, noise sensitivity, and high storage requirement. This paper presents a neighbor prototype selection method based on CCHPSO for intrusion detection. In the model, the prototype selection and feature weight adjustment are performed simultaneously and k-nearest neighbor (KNN) is used as the basic classifier. To deal with large-scale optimization problems, a cooperative coevolving algorithm based on hybrid standard particle swarm and binary particle swarm optimization, which employs the divide-and-conquer strategy, is proposed in this paper. Meanwhile, a fitness function based on the accuracy and data reduction rate is defined in the CCHPSO to obtain a set of appropriate prototypes and feature weights. The KDD99 and NSL datasets are used to assess the effectiveness of the method. The empirical results indicate that the data reduction rate of the proposed method is very high, ranging from 82.32% to 92.01%. Compared with all the data used, the proposed method can not only achieve comparable accuracy performance but also save a lot of storage and computing resources.

1. Introduction

With the continuous development of technology and scale of network, network security incidents have been frequent and the cyber security has already become the focus of all countries in the world. Thousands of companies and agencies around the world were attacked by a ransomware called WannaCry in 2017. The WannaCry has been harmful to 200 thousand computers in more than 150 countries [1]. The attack has had an impact on a large number of institutions around the world. It is necessary to adopt appropriate security technologies such as encryption technology, authentication technology, firewall technology, and antivirus technology [2, 3]. Only under the firewall and the user identity authentication system can not guarantee the cyber security. The intrusion detection system (IDS), as the second security line of active security protection technology, has always been the favor of the researchers.

In different detection environments, IDS can be divided into network intrusion detection system (NIDS) and host intrusion detection system (HIDS). The NIDS employs the network traffic as its data source, and the data source of the HIDS comes from the audit log of the system. Nowadays, the application of machine learning into intrusion detection system has been a trend. There are many intrusion detection systems based on k-nearest neighbors (KNNs), support vector machines (SVMs), extreme learning machines (ELMs), naive Bayes (NBs), decision trees (DTs), and so on. As one of the ten classical algorithms in the field of data mining, KNN is a lazy learning- and instance-based method. Because of the advantages of simple theory, easy implementation, and no need for pretraining, it has been widely used in the field of intrusion detection [4–12]. However, KNN suffers from two major drawbacks [13]. Firstly, the computational complexity and storage consumption are high. Secondly, the algorithm is sensitive to noise samples and isolated objects. With the expansion of network, the network traffic is increasing exponentially. A number of redundant and noise variables exist, which affects the efficiency and accuracy of the detection model [14, 15].

Therefore, there is an urgent need for data reduction techniques.

Data reduction or prototype reduction can be realized by prototype selection (PS) or prototype generation (PG). NN models can be used to guide the search for PS and PG techniques. Prototype selection (PS) technique refers to how to select a set of prototypes from the original training dataset that can represent the training dataset. A minimal set of prototypes can be obtained after prototype selection, so that the performance of a NN model trained on the prototypes is approximately as well as or better than that of a NN model built on the original dataset. PS involves identifying the best subset of the original dataset, and PG concerns creating a new set of objects which can represent the original one. Like feature selection, PS and PG can also be divided into the filter and wrapper [14]. In the filter method, only part of the training dataset is used in the progress of evaluation, while the wrapper method relies on the complete training dataset. The wrapper method can get more higher accuracy although it is more computationally expensive.

The prototype selection mechanisms include condensation, edition, and hybrid [14]. The condensation method is designed to retain samples closer to the decision boundary. The internal points do not affect decision boundaries like boundary points, so internal points can be deleted if the impact on the classification is relatively small. This method can maintain the accuracy of the training set, but the generalization accuracy of the test set may be affected. Since there are fewer boundary points than interior points in most datasets, the condensation method usually has higher compression capabilities. The edition method tries instead to remove the boundary point and keep a smoother decision boundary. The data reduction rate of this method is low. The hybrid approach removes internal and boundary points according to certain criteria and attempts to find a minimal subset that maintains or even increases the precision of generalization in the test dataset. The search directions for prototype selection include incremental, decremental, batch, mixed, and fixed [14]. For the incremental strategy, the size of the selected prototype subset gradually increases from the empty subset. The decremental strategy is just the opposite, and samples that do not meet the standard are gradually deleted. However, it suffers from high complexity over incremental algorithms. The batch method removes all instances that do not meet the criterion at once. The mixed search can iteratively add or remove the instances that meet the criterion. The fixed search is a special case of the mixed strategy. The size of the selected prototypes is fixed, i.e., the number of the additions or removals remains the same.

Most of the prototype selection techniques are combined with 1-NN, mainly because 1-NN is more sensitive to noise samples. This paper also uses 1-NN as the base learner. The heuristic intelligence method for prototype selection has excellent performance both in accuracy and reduction rate. It can improve the classification accuracy of 1-NN and reduce the data by 90% or more [16]. Therefore, researchers have studied combinatorial methods based on heuristic intelligence and nearest neighbor classification [17–24]. In this paper, we use the wrapper method to select prototypes and apply the hybrid selection method and mixed search strategy for prototype reduction. As the feature weighting can enhance the performance of KNN and the feature and instance selection are closely related [25], the feature weighting and prototype selection are simultaneously optimized in this paper. The swarm intelligence heuristic algorithm is a good scheme to do this job [22]. Swarm intelligence (SI), first proposed in 1993, is inspired by animal behaviors such as birds, ants, and fish and is a branch of a population-based heuristic method. SI algorithm has black box optimization capability and does not require prior knowledge of the required field. Particle swarm optimization (PSO), an effective algorithm in swarm intelligence, is commonly used because of its less parameter adjustment and easy implementation.

This paper proposes a method of combining the prototype selection and feature weighting adjustment. We first choose the initial prototypes using the stratification strategy which ensures that every class at least has a prototype as the representative. Then the prototype selection and feature weighting can be combined to improve the performance of KNN. This is obviously an optimization problem and can be solved by the swarm intelligence algorithm. However, with the increase in the dimension of the problem, the performance of many swarm intelligent algorithms will be poor. Thus, a cooperative coevolutionary framework, CCHPSO based on hybrid standard particle swarm and binary particle swarm optimization, is proposed in this paper. It adopts a divide-and-conquer strategy, which can deal with large-scale optimization problems. Finally, two public datasets are used to evaluate the performance of the proposed approach. Experimental results show that the framework of using the prototype selection method gives comparable accuracy than that of using all datasets. This method can also save a lot of storage and computing resources which has a wide range of application prospect in the era of big data.

The paper is organized as follows: Section 2 gives the related works. The background techniques are listed in Section 3. Section 4 reports the method this paper proposed. The experimental results are presented in Section 5. Finally, some concluding remarks are given in Section 6.

2. Related Work

Because of its key advantage of simplicity and high precision, the KNN model and its variants have been widely used in the field of intrusion detection. Aburomman and Ibne Reaz [4] combined six k-nearest neighbor (KNN) models and six support vector machine (SVM) experts using PSO. They showed that the method has better accuracy than weighted majority voting (WMV). Meng et al. [5] proposed an enhanced filter method of misuse intrusion detection, and KNN is adopted as the false alarm filter. They showed the performance of the signature-based IDS has been enhanced. Meng et al. [6] developed an alert verification, and KNN is used to filter out unwanted alarms. They showed the alarm filter can effectively filter out plenty of alarms. Tsai and Lin [7] proposed a method named "TANN." The training dataset is divided into five categories by k-means, and new features

of training dataset are formed by the area of the triangle which connects any two cluster centers and one of the original training samples. Finally, the KNN classifier is used to detect attacks based on the new dataset. Lin et al. [8] presented the CANN model which is also a new feature representation approach. It is worth mentioning that KNN is also selected to do the final classification. The above two papers give new feature representation approaches, and KNN is used as a benchmark for all the other classifiers. Sharafaldin et al. [9] produced a new type of network data which includes normal type and seven types of attacks. The machine learning algorithms were evaluated over the dataset and they reported that the KNN, random forest, and ID3 have good performance. Kuttranont et al. [10] showed that KNN is one of the promising approaches. Since big data exerts great pressure on machine learning algorithms, they proposed the implementation of KNN on GPU. Chen et al. [11] proposed a compressed model using MapReduce. KNN and SVM are employed to evaluate the performance of the compressed model. KNN has been widely applied in the above works; however, the prototype selection under the guidance of KNN is not considered.

There are many methods proposed about the prototype selection or prototype generation. Most of them use divide-and-conquer and merging strategies to select or generate new artificial samples. Haro-García and García-Pedrajas [26] proposed a divide-and-conquer recursive approach for very large problems. The method divides the original training dataset into small subsets where the prototype selection is applied. Then, the selected prototypes are rejoined in a new training dataset, and the above procedure is repeated again. Triguero et al. [27] developed a MapReduce-based framework named MRPR to distribute the functioning of the prototype reduction algorithms. The authors offer a MapReduce paradigm that gives a simple and efficient environment to parallelize the prototype reduction computation. How to produce the prototypes is not the focus of this article. Escalante et al. [28] introduced a novel approach named PGGP of PG methods. Highly effective prototypes are built based on genetic programming in which many training samples are combined through arithmetic operators. The authors showed that the method outperforms other PG approaches. Paredes and Vidal [29] proposed a new gradient descent method named learning prototype and distance (LDP). A small number of prototypes are selected, and then the position of the prototypes and their weights have been iteratively adjusted.

Some heuristic algorithms have been applied to prototype selection or prototype generation. Nanni and Lumini [18] proposed a prototype reduction method based on particle swarm optimization. The algorithm flow is similar to the processing of the random subspace in the random forest. During the training phase, the prototype generation is repeated many times, then each of the training model is used to classify each test sample, and finally the classification results are combined by the majority vote rule. Triguero et al. [19] reported a prototype generation methodology about positioning adjustment. Differential evolution is used to optimize the positioning of prototypes in nearest neighbor classification. Rezaei and Nezamabadi-Pour [20] applied the gravitational search algorithm (GSA) to generate prototypes for nearest neighbor classification. The initial objects are extracted using the stratification strategy. Derrac et al. [21] presented an approach which integrates instance selection, feature weighting, and instance weighting schemes into one. They reported that the approach can enhance the results of the 1-NN classifier. Pérez-Rodríguez et al. [22] proposed a framework of combining instance and feature selection and weighting to improve the performance of the data mining methods. Differential evolution and a binary CHC genetic algorithm are adopted to perform the weighting adjustment and selection, and 1-NN is used as the classifier. Escalante et al. [23] introduced a multiobjective evolutionary algorithm based on NSGA-II for prototype generation. Kardan et al. [24] proposed a novel hybrid approach named BBO-KNN. The biogeography-based optimization (BBO) is used to optimize the input features, feature weight, and parameter K of KNN rule.

3. Background

3.1. k-Nearest Neighbor (KNN). k-Nearest neighbor (KNN) is a simple and effective classification technique. Unlike SVM, KNN can directly deal with multiclass problems and has a wide range of applications.

KNN is a supervised classification algorithm. The training samples are expressed as (x_i, y_i), where $x_i = (x_{i1}, x_{i2}, \ldots, x_{i\,D}) \in R^D$, D represents the number of features, and y_i represents the label. For a test sample, its label will be determined by its peripheral training samples, that is, it will be predicted by the majority of the labels of the training samples around it. Generally, Euclidean distance is used to measure the similarity between the samples, which is defined as follows:

$$d\left(x_i, x_j\right) = \sqrt{\sum_{r=1}^{D} \left(x_{ir} - x_{jr}\right)^2}, \tag{1}$$

where $d(x_i, x_j)$ denotes the Euclidean distance between x_i and x_j and x_{ir} represents the r-th feature of the i-th sample.

The parameter K in KNN represents the number of neighbor samples around the query sample, and the selection of K is important for the performance of the KNN.

3.2. Prototype Selection. Prototype selection, as a data preprocessing step, can remove the noise and abnormal points and reduce the size of the training set. Let TR represents the original training dataset (including the noise and redundant information). Select TS from TR whose size is less than that of TR, yet the accuracy based on TS is almost the same as that based on TR. TS takes the place of TR as the benchmark data for training, thus saving the storage space and reducing the computational complexity.

3.3. Particle Swarm Optimization. In PSO, every particle in a D-dimensional space represents a potential solution. The particle has two properties, including the velocity and

Input: the algorithm parameters
Output: the global optimal result.
(1) **Repeat**
(2) **for** each swarm j
(3) **for** each particle i
(4) **If** $f(b(j, P_j \cdot x_i)) < f(b(j, P_j \cdot y_i))$ **then** $P_j \cdot y_i = P_j \cdot x_i$ **end if**;
(5) **If** $f(b(j, P_j \cdot y_i)) < f(b(j, P_j \cdot y'))$ **then** $P_j \cdot y' = P_j \cdot y_i$ **end if**;
(6) **end for**
(7) Perform the position and velocity update using (2), (3), or (4)
(8) **end for**
(9) **until** termination is met;

Algorithm 1: The CPSO-S_K: for a particle I, $P_j \cdot x_i$, $P_j \cdot y_i$, and $P_j \cdot y'$ denote the position, the personal best position, and the global best particle of the swarm P_j, f represents the fitness function, and b (j, z) return the global solution where the j-th position is z.

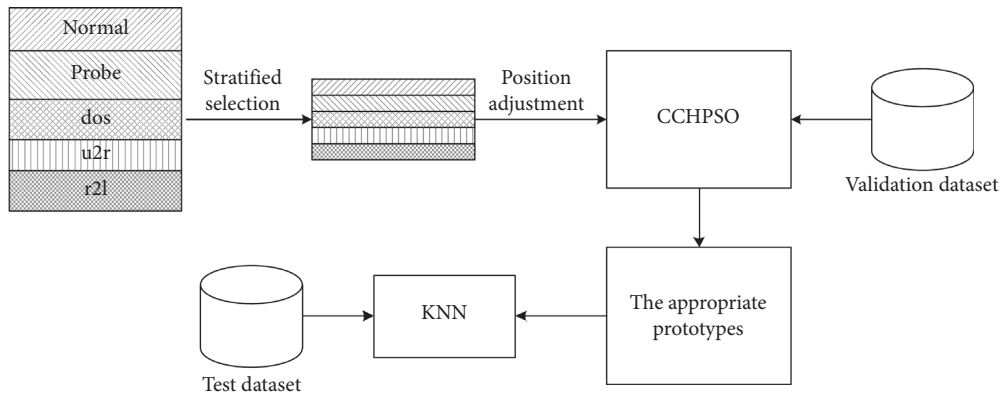

Figure 1: Architecture of the proposed model for intrusion detection.

position. The fitness is also an important property which is the evaluation of a particle. The optimal position ($p$best) and the global optimal position ($g$best) can be simultaneously perceived by every particle. The velocity and position are updated as follows:

$$V_i^t = \omega_{\text{pso}} \times V_i^{t-1} + c_1 \times \text{rand}(\,) \times \left(p\text{best}_i^t - X_i^t\right) + c_2 \times \text{rand}(\,) \times \left(g\text{best}_i^t - X_i^t\right), \quad (2)$$

$$X_i^t = X_i^{t-1} + V_i^{t-1}, \quad (3)$$

where V_i^t and X_i^t indicate the velocity and position of the i-th particle in the t-th iteration, $p\text{best}_i^t$ and $g\text{best}_i^t$ represent the previous best position of the i-th particle and the global optimal position until iteration t, w_{pso} is the constriction coefficient, c_1 and c_2 are acceleration coefficients, and rand( ) is a random number which is uniformly distributed in [0, 1].

The discrete binary version of PSO (BPSO) was designed by Kennedy and Eberhart [30]. In BPSO, the position is made of a binary string. Compared with the standard particle swarm, only the position update rules are different which is as follows:

$$X_i^t = \begin{cases} 1, & \text{if rand}(\,) \le s\left(V_i^t\right), \\ 0, & \text{otherwise}, \end{cases} \quad (4)$$

where V_i^t is mapped to interval [0, 1] by sigmoid function $s(V_i^t) = 1/(1 + e^{-V_i^t})$.

To solve large-scale optimization problems, a cooperatively coevolving PSO, CPSO-S_K, was proposed by VandenBergh and Engelbrecht [31]. The idea is very simple, and the divide-and-conquer strategy is employed. The solution can be split into L subcomponents, and each will evolve in the pattern of the PSO. The final global optimal position is composed of the optimal solution of each swarm. The pseudocode of CPSO-S_K is shown in Algorithm 1.

4. The Proposed Method and Analysis

4.1. Stratification Strategy. The initial population must ensure the diversity of the classes of samples. Specifically, we select the initial prototypes from the original dataset using the stratification strategy, which is extracting the prototypes randomly in a certain proportion from different layers of the original dataset. The stratified ratio can be adjusted flexibly.

4.2. Feature Weighting Adjustment. In practical problems, the importance of different features is often different when measuring the similarity between samples. The solution is to give each feature a different weight to represent the importance of the feature. Formula (1) can be improved as

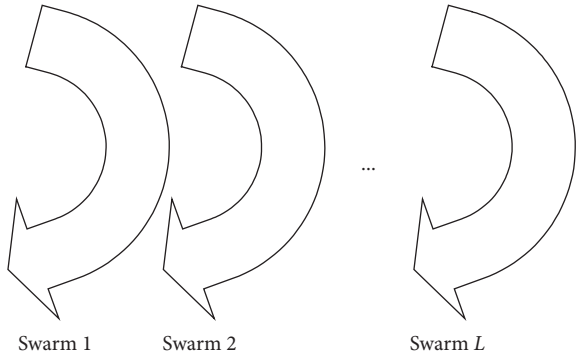

Figure 2: The iterative process of the CCHPSO.

Table 1: Particle representation.

Instance mask						Feature weight			
0	1	—	0	0	1	0.79	0.8	—	0.21

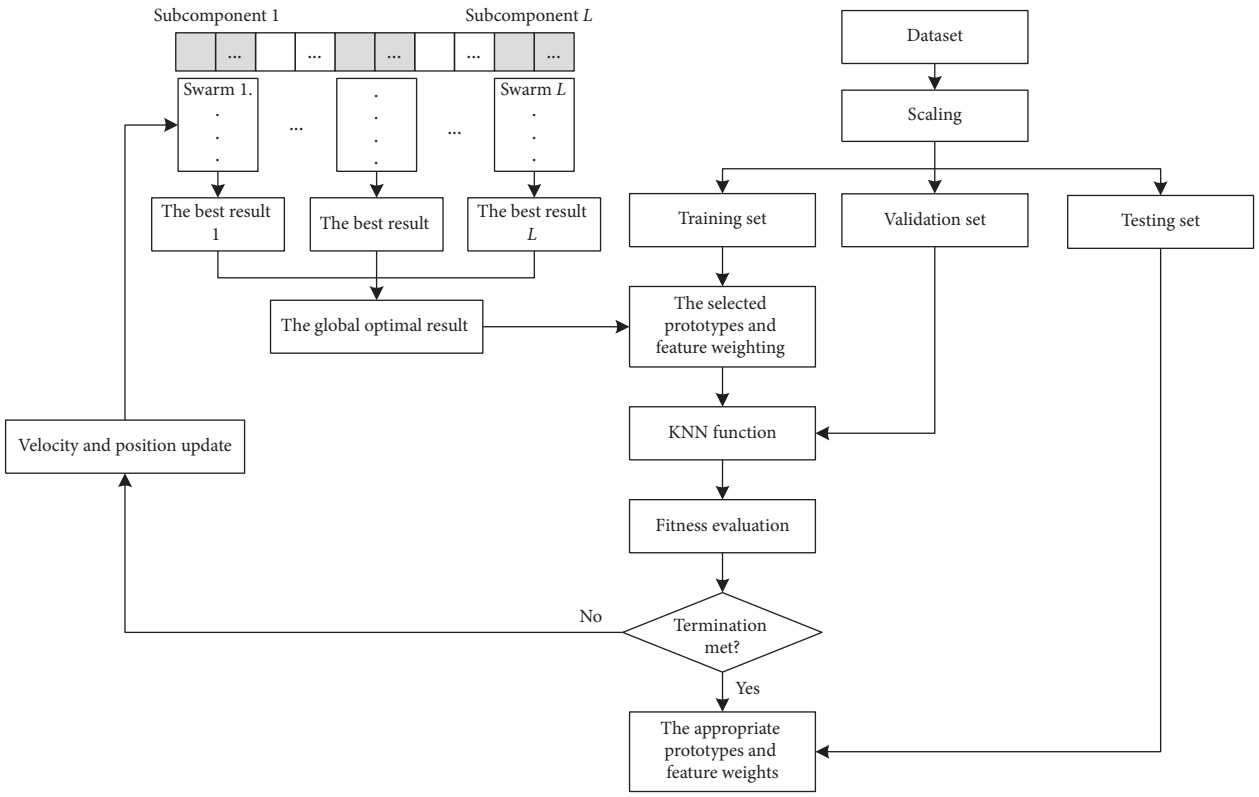

Figure 3: Flow diagram of the proposed CCHPSO-based prototype selection and feature weighting adjustment for KNN in intrusion detection.

$$\mathrm{Fd}\left(x_i, x_j\right) = \sqrt{\sum_{r=1}^{D} \mathrm{Fw}_r \left(x_{ir} - x_{jr}\right)^2}, \tag{5}$$

where $\mathrm{Fd}(x_i, x_j)$ denotes the Euclidean distance between x_i and x_j that takes into account the feature weighting and Fw_r represents the feature weighting of the r-th feature.

4.3. Block Diagram of the CCHPSO-KNN. This section describes the proposed method for the intrusion detection system. The overall process of the proposed model is illustrated in Figure 1. As the initial dataset is huge, the training dataset will be obtained by the stratification strategy, i.e., ensuring that each class has some prototypes. There may be redundancy or noise points in the data, and CCHPSO is used to make prototype selection and feature weighting

Input: training, validation, and testing datasets with labels, KNN as the main classifier, CCHPSO algorithm
Output: testing accuracy (acc), DR, FPR, and confusion matrix.
(1) Training:
(2) Obtain the training, validation, and testing datasets by the stratification strategy
(3) **repeat**
(4) **for** each swarm
(5) **for** each particle
(6) fitness = KNN (pop, train scale, train label, validation scale, and validation label);
(7) update the local and global Sol;
(8) **end for**
(9) Perform position and velocity updates using (2), (3), or (4)
(10) **end for**
(11) **until** termination is met;
(12) Obtain the appropriate prototypes and feature weights according to the global optimal Sol.
(13) Testing:
(14) [testing accuracy, confusion matrix] = KNN (Sol, prototype data, prototype label, test scale, test label);

Algorithm 2: The proposed method.

Table 2: Features of the KDD99 and NSL.

Feature representation	Feature name
F_1	Duration
F_2	protocol_type
F_3	Service
F_4	Flag
F_5	src_bytes
F_6	dst_bytes
F_7	Land
F_8	wrong_fragment
F_9	Urgent
F_{10}	Hot
F_{11}	num_failed_logins
F_{12}	logged_in
F_{13}	num_compromised
F_{14}	root_shell
F_{15}	su_attempted
F_{16}	num_root
F_{17}	num_file_creations
F_{18}	num_shells
F_{19}	num_access_files
F_{20}	num_outbound_cmds
F_{21}	is_hot_login
F_{22}	is_guest_login
F_{23}	Count
F_{24}	srv_count
F_{25}	serror_rate
F_{26}	srv_serror_rate
F_{27}	rerror_rate
F_{28}	srv_rerror_rate
F_{29}	same_srv_rate
F_{30}	diff_srv_rate
F_{31}	srv_diff_host_rate
F_{32}	dst_host_count
F_{33}	dst_host_srv_count
F_{34}	dst_host_same_srv_rate
F_{35}	dst_host_diff_srv_rate
F_{36}	dst_host_same_src_port_rate
F_{37}	dst_host_srv_diff_host_rate
F_{38}	dst_host_serror_rate
F_{39}	dst_host_srv_serror_rate
F_{40}	dst_host_rerror_rate
F_{41}	dst_host_srv_rerror_rate

Table 3: Confusion matrix.

	Predicted attack	Predict normal
Attack	(TP)	(FN)
Normal	(FP)	(TN)

adjustment. Finally, the KNN model will be used to classify the test dataset based on the generated prototypes and feature weights. The dataset we used will be divided into three parts: the training dataset, validation dataset, and testing dataset. The training dataset will be used to produce the prototypes and the feature weights, the validation dataset is employed to validate the feasibility of the selected prototypes and feature weights during the training process, and the generated prototype and feature weights will be used to test the test dataset in the last step.

In the first stage, CCHPSO is used to select the instance subset and feature weight. The D-dimensional object vector is decomposed into L subcomponents illustrated in Figure 2; i.e., each of the L-subcomponents corresponds to a swarm which has s-dimensions selected from the D-dimensional object vector ($D = L * s$). The arrow in Figure 2 indicates the iterative process of each swarm which will output a best result after it evolves. The iterative process of the cooperatively coevolving algorithm is just like unlocking a suitcase's password lock. The global optimal results can be obtained by combing the results evolved from different subcomponents.

In particular, the particle and the fitness function need to be defined first. A particle is comprised of two parts including the instance mask and feature weight. The structure of a particle is shown in Table 1. The first half of the table is a binary string which represents the instance is selected or not, and the second half of the table denotes the feature weights. Suppose there are n instances and m features, and thus there are a total of $n + m$ bits of the particle.

In this model, the high classification accuracy and the few instances are the criteria to design a fitness function. Thus, the fitness function can be defined as

Table 4: Experimental results with the public datasets when $K = 1$ (%).

Datasets	Methods	Training acc	Testing acc	DR	FPR	*R*rate
Kdd	CCHPSO	98.24	97.07	99.01	2.25	92.01
	No selection	98.61	97.70	99.32	3.5	0
	No weighting	98.40	96.72	98.82	3.5	91.85
Nsl	CCHPSO	95.74	90.86	97.42	10	87.26
	No selection	97.70	95.75	98.14	7.50	0
	No weighting	95.81	90.66	96.78	8.25	86.97

Table 5: Experimental results with the public datasets when $K = 3$ (%).

Datasets	Methods	Training acc	Testing acc	DR	FPR	*R*rate
Kdd	CCHPSO	97.15	96.30	98.61	3.75	89.97
	No selection	98.05	96.72	98.53	4	0
	No weighting	96.66	96.16	98.12	3	89.83
Nsl	CCHPSO	94.07	89.61	96.24	9.25	83.20
	No selection	96.93	94.63	98.24	11.25	0
	No weighting	94.36	90.93	97.38	6.75	85.02

Table 6: Experimental results with the public datasets when $K = 5$ (%).

Datasets	Methods	Training acc	Testing acc	DR	FPR	*R*rate
Kdd	CCHPSO	96.46	95.89	98.51	4.75	88.75
	No selection	97.21	95.96	98.32	4	0
	No weighting	96.05	95.47	98.02	5	89.41
Nsl	CCHPSO	92.53	88.91	96.42	10.5	82.32
	No selection	96.51	94.63	98.23	10	0
	No weighting	93.37	90.00	96.85	8.25	82.65

$$\text{fitness} = \omega_1 \times \text{acc} + \omega_2 \times R\text{rate}, \tag{6}$$

where "acc" denotes the classification accuracy based on the current chosen instances, *R*rate represents the reduction rate, w_1 is the weight for the classification accuracy, and w_2 is the weight for the instance selection evaluation. The flow diagram of the proposed method is shown in Figure 3, and the pseudocodes for the proposed method are shown in Algorithm 2.

5. Experiments

5.1. Dataset Used for Experiments. The KDD99 [32] and NSL-KDD [33] were used to demonstrate the generalization ability of the proposed method. Over the years, KDD99 is still recognized as the standard dataset in the field of IDS. Each network connection in the KDD99 and NSL-KDD dataset is described by 41 features shown in Table 2. The types of samples are divided into five categories, including Normal, Probe, DoS, U2R, and R2L. The NSL is more demanding on the IDS method in which duplicate records were removed so that the sample types can reach a balance.

5.2. Evaluation. The accuracy (Acc), detection rate (DR), and false-positive rate (FPR) are used to assess the performance of the intrusion detection method. The above indexes can be obtained by the confusion matrix shown in Table 3, and Acc, DR, and FPR can be expressed as follows.where TP represents the number of attacks correctly recognized, FP represents the number of normal records predicted as attack, FN denotes the number of attacks recognized as normal, and TN represents the number of normals correctly classified.

$$\begin{aligned} \text{accuracy} &= \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}, \\ \text{DR} &= \frac{\text{TP}}{\text{TP} + \text{FN}}, \\ \text{FPR} &= \frac{\text{FP}}{\text{FP} + \text{TN}}, \end{aligned} \tag{7}$$

5.3. Experimental Results. The population size for each swarm and the number of iterations are set to be 50 and 200, respectively. Under the dual factors of acc and *R*rate, CCHPSO can iteratively select prototypes and adjust the feature weighting. w_1 and w_2 in the fitness function are set to be 0.9 and 0.1, respectively. All experiments were run on the Matlab R2012a platform equipped with a 2.4 GHz CPU and 32 GB of RAM.

The selection of parameter *K* directly affects the output of KNN. Since the individual is comprised of two parts including the instance mask and feature weight, the experimental results of without considering the prototype selection (no selection)

and without considering the feature weights adjustment (no weighting) under different K values are analyzed. Tables 4–6 show the experimental results of CCHPSO, the no selection method and the no weighting method, respectively, on KDD and NSL datasets when $K = 1$, 3, and 5. The evaluation criteria include the training acc, testing acc, DR, FPR, and Rrate. All results are averaged by ten experiments.

Since the NSL is more demanding on the algorithms, the experimental results based on KDD are generally better than those based on NSL. The accuracy of CCHPSO is 97.07% and 90.86%, respectively, and the false alarm rate is 2.25% and 10% when $K = 1$. It can be concluded that the experimental results are more stable and effective when $K = 1$.

Overall, the no selection method performs best. Because the feature weight is optimized and there is no prototype selection, the prototype data remains intact. It also shows that the reduction rate of the prototype selection using CCHPSO is very high, ranging from 82.32% to 92.01%. From Tables 4–6, we can see that when the data are reduced by about 90%, and the accuracy and other indicators are not greatly affected.

6. Conclusions

The machine learning algorithms are seriously challenged by large datasets, and KNN is one of the most relevant algorithms in machine learning. In this paper, a neighbor prototype selection method based on CCHPSO has been proposed for intrusion detection. The KNN is chosen as the base classier, and the PSO, which can be implemented easily and has few parameters to tune, is used to select prototypes and adjust feature weighting. Moreover, to deal with large-scale optimization problems, a cooperatively coevolving method based on hybrid standard PSO and binary PSO, which employs the divide-and-conquer strategy, is employed. The training samples are generated via the stratification strategy which can ensure the diversity of the classes of samples. Finally, the KNN model is used to classify the test dataset based on the generated prototypes and feature weights. The experiments were conducted on two public datasets to evaluate the effectiveness of the CCHPSO and no selection and no weighting methods. The experimental results show that the reduction rate of the prototype selection using CCHPSO is very high, reaching 92.01%. It can also be concluded that when the data are reduced by about 90%, the accuracy and other indicators are not greatly affected. To advance the execution efficiency, the next step is to improve the model based on GPU parallel computing.

Data Availability

The data supporting this article are from previously reported studies and datasets, which have been cited.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Key R&D Program of China (2017YFB0802803) and the National Natural Science Foundation of China (61602052).

References

[1] J. M. Ehrenfeld, "WannaCry, cybersecurity and health information technology: a time to act," *Journal of Medical Systems*, vol. 41, no. 7, p. 104, 2017.

[2] M. Zhang, Y. Zhang, Y. Jiang, and J. Shen, "Obfuscating EVES algorithm and its application in fair electronic transactions in public cloud systems," *IEEE System Journal*, vol. 13, no. 2, pp. 1478–1486, 2019.

[3] M. Zhang, Y. Yao, Y. Jiang, B. Li, and C. Tang, "Accountable mobile e-commerce scheme in intelligent cloud system transactions," *Journal of Ambient Intelligence and Humanized Computing*, vol. 9, no. 6, pp. 1889–1899, 2018.

[4] 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.

[5] W. Meng, W. Li, and L.-F. Kwok, "EFM: enhancing the performance of signature-based network intrusion detection systems using enhanced filter mechanism," *Computers & Security*, vol. 43, no. 6, pp. 189–204, 2014.

[6] W. Meng, W. Li, and L. F. Kwok, "Design of intelligent KNN-based alarm filter using knowledge-based alert verification in intrusion detection," *Security & Communication Networks*, vol. 8, no. 18, pp. 3883–3895, 2015.

[7] C.-F. Tsai and C.-Y. Lin, "A triangle area based nearest neighbors approach to intrusion detection," *Pattern Recognition*, vol. 43, no. 1, pp. 222–229, 2010.

[8] W.-C. Lin, S.-W. Ke, and C.-F. Tsai, "CANN: an intrusion detection system based on combining cluster centers and nearest neighbors," *Knowledge-Based Systems*, vol. 78, no. 1, pp. 13–21, 2015.

[9] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proceedings of the 4th International Conference on Information Systems Security and Privacy*, Funchal, Portugal, January 2018.

[10] P. Kuttranont, K. Boonprakob, C. Phaudphut et al., "Parallel KNN and neighborhood classification implementations on GPU for network intrusion detection," *Journal of Telecommunication, Electronic and Computer Engineering*, vol. 9, no. 2, pp. 29–33, 2017.

[11] T. Chen, X. Zhang, S. Jin, and O. Kim, "Efficient classification using parallel and scalable compressed model and its application on intrusion detection," *Expert Systems with Applications*, vol. 41, no. 13, pp. 5972–5983, 2014.

[12] A.-F. Farhan, Z. M. Dahalin, and J. Shaidah, "Distributed and cooperative hierarchical intrusion detection on MANETs," *International Journal of Computer Applications*, vol. 12, no. 5, pp. 32–40, 2010.

[13] U. Garain, "Prototype reduction using an artificial immune model," *Pattern Analysis & Applications*, vol. 11, no. 3-4, pp. 353–363, 2008.

[14] S. Garcia, J. Derrac, J. R. Cano, and F. Herrera, "Prototype selection for nearest neighbor classification: taxonomy and empirical study," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 3, pp. 417–435, 2012.

[15] I. Triguero, J. Derrac, S. Garcia, and F. Herrera, "A taxonomy and experimental study on prototype generation for nearest

neighbor classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 1, pp. 86–100, 2012.

[16] N. Verbiest, S. Vluymans, C. Cornelis, N. García-Pedrajas, and Y. Saeys, "Improving nearest neighbor classification using ensembles of evolutionary generated prototype subsets," *Applied Soft Computing*, vol. 44, pp. 75–88, 2016.

[17] W. Hu and Y. Tan, "Prototype generation using multi-objective particle swarm optimization for nearest neighbor classification," *IEEE Transactions on Cybernetics*, vol. 46, no. 12, pp. 2719–2731, 2016.

[18] L. Nanni and A. Lumini, "Particle swarm optimization for prototype reduction," *Neurocomputing*, vol. 72, no. 4–6, pp. 1092–1097, 2009.

[19] I. Triguero, S. García, and F. Herrera, "Differential evolution for optimizing the positioning of prototypes in nearest neighbor classification," *Pattern Recognition*, vol. 44, no. 4, pp. 901–916, 2011.

[20] M. Rezaei and H. Nezamabadi-Pour, "Using gravitational search algorithm in prototype generation for nearest neighbor classification," *Neurocomputing*, vol. 157, pp. 256–263, 2015.

[21] J. Derrac, I. Triguero, S. Garcia, and F. Herrera, "Integrating instance selection, instance weighting, and feature weighting for nearest neighbor classifiers by coevolutionary algorithms," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 5, pp. 1383–1397, 2012.

[22] J. Pérez-Rodríguez, A. G. Arroyo-Peña, and N. García-Pedrajas, "Simultaneous instance and feature selection and weighting using evolutionary computation: proposal and study," *Applied Soft Computing*, vol. 37, pp. 416–443, 2015.

[23] H. J. Escalante, M. Marin-Castro, A. Morales-Reyes et al., "MOPG: a multi-objective evolutionary algorithm for prototype generation," *Pattern Analysis and Applications*, vol. 20, no. 1, pp. 33–47, 2017.

[24] A. A. Kardan, A. Kavian, and A. M. Esmaeili, "Simultaneous feature selection and feature weighting with K selection for KNN classification using BBO algorithm," in *Proceedings of the 5th Information and Knowledge Technology*, pp. 349–354, IEEE, Shiraz, Iran, May 2013.

[25] N. García-Pedrajas, A. de Haro-García, and J. Pérez-Rodríguez, "A scalable approach to simultaneous evolutionary instance and feature selection," *Information Sciences*, vol. 228, no. 7, pp. 150–174, 2013.

[26] A. D. Haro-García and N. García-Pedrajas, "A divide-and-conquer recursive approach for scaling up instance selection algorithms," *Journal Data Mining and Knowledge Discovery*, vol. 18, no. 3, pp. 392–418, 2009.

[27] I. Triguero, D. Peralta, J. Bacardit, S. Garcrd, and F. Herrera, "MRPR: a MapReduce solution for prototype reduction in big data classification," *Neurocomputing*, vol. 150, no. 150, pp. 331–345, 2015.

[28] H. J. Escalante, M. Graff, and A. Morales-Reyes, "PGGP: prototype generation via genetic programming," *Applied Soft Computing*, vol. 40, pp. 569–580, 2016.

[29] R. Paredes and E. Vidal, "Learning prototypes and distances: a prototype reduction technique based on nearest neighbor error minimization," *Pattern Recognition*, vol. 39, no. 2, pp. 180–188, 2006.

[30] J. Kennedy and R. Eberhart, "A discrete binary version of the particle swarm algorithm," in *Proceedings of the IEEE Conference on Systems, Man, and Cybernetics*, pp. 4104–4108, Orlando, FL, USA, October 1997.

[31] F. VandenBergh and A. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 225–239, 2004.

[32] Archibe, U. K. KDD Cup 1999 Data, 1999, http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

[33] Archibe, U. K. NSL Data, 2006, http://nsl.cs.unb.ca/NSL-KDD.