EURASIP Journal on Wireless Communications and Networking

RESEARCH Open Access

A novel two-phase cycle algorithm for effective cyber intrusion detection in edge computing

Yiguang Gong*, Yunping Liu and Chuanyang Yin

*Correspondence: yiguang-gong@nuist.edu.cn
School of Automation, Nanjing University of Information Science and Technology, Nanjing 210000, China

Abstract
Edge computing extends traditional cloud services to the edge of the network, closer to users, and is suitable for network services with low latency requirements. With the rise of edge computing, its security issues have also received increasing attention. In this paper, a novel two-phase cycle algorithm is proposed for effective cyber intrusion detection in edge computing based on a multi-objective genetic algorithm (MOGA) and modified back-propagation neural network (MBPNN), namely TPC-MOGA-MBPNN. In the first phase, the MOGA is employed to build a multi-objective optimization model that tries to find the Pareto optimal parameter set for MBPNN. The Pareto optimal parameter set is applied for simultaneous minimization of the average false positive rate (Avg FPR), mean squared error (MSE) and negative average true positive rate (Avg TPR) in the dataset. In the second phase, some MBPNNs are created based on the parameter set obtained by MOGA and are trained to search for a more optimal parameter set locally. The parameter set obtained in the second phase is used as the input of the first phase, and the training process is repeated until the termination criteria are reached. A benchmark dataset, KDD cup 1999, is used to demonstrate and validate the performance of the proposed approach for intrusion detection. The proposed approach can discover a pool of MBPNN-based solutions. Combining these MBPNN solutions can significantly improve detection performance, and a GA is used to find the optimal MBPNN combination. The results show that the proposed approach achieves an accuracy of 98.81% and a detection rate of 98.23% and outperform most systems of previous works found in the literature. In addition, the proposed approach is a generalized classification approach that is applicable to the problem of any field having multiple conflicting objectives.

Keywords: Two-phase cycle algorithm, Edge computing, Cyber intrusion detection

1 Introduction

With the advancement of edge computing, the number of edge services running on mobile devices is growing explosively [1]. Edge computing can reduce processing times and improve application performance, but it also presents some new challenges. Privacy is a serious issue in edge computing as user data is collected, processed, transmitted and shared over edge nodes [2]. The private information (i.e., location, service type, etc.) in the services is prone to release during service offloading [3], so it is imperative to improve privacy security in edge computing [4]. Encrypting sensitive data before outsourcing is conducive to privacy security, but it will require additional data uploads [5], while edge nodes are always expected to conserve energy by uploading the minimum possible amount of data, and it is meaningful to find a mechanism to consume as little energy as possible for data uploading [6]. The explosive growth and variety of information available on edge nodes frequently overwhelm users; recommender systems are a promising way for users to quickly find the valuable information that they are interested in from massive data [7–9]. In addition to these challenges, edge computing introduces a scale of cyber security challenges that regular data center operators may not be accustomed to dealing with. An intrusion, which is one of the main cyber security challenges, is any set of actions intended to compromise the confidentiality, integrity, or availability of a resource [10]. Cyber intrusions are prevalent, increasingly sophisticated, and are adept at hiding from detection [11]. To counteract ever-evolving threats, the network-based intrusion detection system (NIDS) has been considered to be one of the most promising methods.

Intrusion detection techniques have become a significant topic of research in recent years. Many researchers propose different algorithms in different categories, such as artificial neural networks (ANNs) [12], SVM [13], k-nearest neighbor [14], random forest [15], deep learning approaches [16], Bayesian approaches [17] and decision trees [18]. As a result, the performances of detection techniques are becoming better and stronger.

The artificial neural network (ANN), the computing paradigm that mimics the way neural systems of the human brain work, is widely used in cyber intrusion detection. Hodo et al. [19] present a multi-level perceptron, a type of supervised artificial neural network (ANN) to detect distributed denial of service (DDoS/DoS) attacks. The experimental results demonstrate 99.4% accuracy and can successfully detect various DDoS/DoS attacks. Yin et al. [20] propose a deep learning approach for intrusion detection using recurrent neural networks (RNN-IDS). The experimental results show that RNN-IDS is suitable for modeling a classification model with high accuracy. Naseer and Saleem [21] propose a deep convolutional neural network (DCNN)-based intrusion detection system (IDS). The experimental results of the proposed DCNN-based IDS show the promising results for real world application in anomaly detection systems. Benmessahel et al. [22] present an evolutionary neural network (ENN), which is a combination of ANN and evolutionary algorithm (EA), and experiments show that this approach is effective for cyber intrusion detection. Arul Anitha and Arockiam [23] propose an artificial neural network-based IDS (ANNIDS) technique based on a multilayer perceptron (MLP) to detect the attacks initiated by destination oriented direct acyclic graph information solicitation (DIS) attack and version attack in IoT environments. Sun and Lyu [24] use the LSTM neural network with long and short memory function to train the KDD99 dataset, and identify the DOS according to the trained model. This is a research process pertaining to planned adjustment of the hyperparameters to find the optimal solution after processing the data. Shenfield et al. [25] present a novel approach to detecting malicious cyber traffic using artificial neural networks suitable for use in deep packet inspection-based IDS. The presented results show that this novel classification approach is capable of detecting shell code with extremely high accuracy and minimal numbers of false identifications. Amruta and Talha [26] present a Denial of Service Attack Detection system using ANN for wired LANs. The proposed ANN classifier yields 96% accuracy for their training dataset. Most of the systems have produced promising classification accuracy.

ANN has the ability to approximate an arbitrary function mapping and learn from examples, much like the human brain. In many cases, ANN surpasses the conventional statistical method for the classification task in various fields of application [27]. However, designing an ANN is a difficult process. Its performance depends on the optimization of various design parameters such as choosing an optimal number of hidden nodes, suitable learning algorithm, learning rate and initial value of the weights, and some objectives conflict with each other, such as accuracy and complexity. Therefore, multi-objective optimization (MOO) is considered to be a more realistic approach to the design of ANN than the single-objective approach [28]. In addition, ANN has some shortcomings such as slow convergence speed, entrapment in local optima and unstable network structure [29]. In contrast, the genetic algorithm (GA) exhibits its characteristics of global search and quick convergence ability and is the most widely used technique in data mining and knowledge discovery [30]. On the other hand, the method of Pareto optimality has been widely used in MOO [31]. It offers a pool of non-inferior individual solutions and ensemble solutions instead of a single optimum, which accordingly provides more degrees of freedom in selection of proper solutions. The multi-objective genetic algorithm (MOGA) [32] and the non-dominated sorting genetic algorithm (NSGA II [33], NSGA III [34]) are two examples of GA-based MOO methods that apply the concept of Pareto optimality.

Some multi-objective genetic algorithm (MOGA)-based approaches are proposed for effective intrusion detection based on benchmark datasets. Elhag et al. [35] propose a multi-objective evolutionary fuzzy system that can be trained using different metrics. The system obtains more accurate solutions and allows the final user to decide which solution is better suited for the current network characteristics. Stehlík et al. [36] propose multi-objective evolutionary algorithms (NSGA-II and SPEA2 [37]) for intrusion detection parameterization, with a focus on the impact of the evolutionary algorithm (and its parameters) on the optimality of the found solutions, the speed of convergence and the number of evaluations. Kumar and Kumar [30] propose the three-phase MOGA-based algorithm Micro Genetic Algorithm2 (AMGA2) [38], which considers conflicting objectives simultaneously, such as detection rate of each attack type, error rate, accuracy and diversity. In the first phase, a Pareto front of non-inferior individual solutions is approximated. In the second phase, the entire solution set is further refined, and another improved Pareto front of ensemble solutions is approximated over that of individual solutions. In the third phase, a combined method like the majority voting method is used to fuse the predictions of individual solutions for determining predictions of ensemble solutions. The experiments conducted on two benchmark datasets demonstrate that the proposed approach can discover individual solutions and ensemble solutions for intrusion detection.

This paper aims to develop a novel two-phase cycle training algorithm for intrusion detection. The MOGA-based approach is used to find the Pareto optimal parameter set for the neural networks. A modified back-propagation neural network (MBPNN) set is created based on the Pareto optimal parameter set and trained to find a more optimal parameter set locally. The proposed approach can discover a pool of MBPNN-based solutions to detect the intrusions accurately. The approach proposed in this paper has been published by the international conference on ML4CS 2020 [39]. Based on the conference paper, this report is mainly expanded as follows: A genetic algorithm is used to find the optimal combination solution set for prediction, instead of the manual selection method used in the conference paper, and the genetic algorithm manifests better performance. Some experimental datasets are added, all experiments are repeated, and the corresponding described text, figure and table of the experimental results are updated.

The rest of this paper is organized as follows: Sect. 2 presents an overview of the proposed methodology. The experimental results and discussion are presented in Sect. 3. Finally, the concluding remarks of the study are provided in Sect. 4.

2 Materials and methods

2.1 The proposed approach

TPC-MOGA-MBPNN includes training session, testing session and the combined method, as described below.

2.1.1 The training session

The training session as illustrated in Fig. 1 is implemented by TPC-MOGA-MBPNN. In the first phase, a MOGA tries to find the Pareto optimal parameter set for the neural networks. The MOGA considers average false positive rate (Avg FPR), mean squared error (MSE) and negative average true positive rate (Avg TPR) on the training dataset as the objectives to be minimized. Meanwhile, the weights, biases and gain factors of neural networks become the genotype to be evolved simultaneously by MOGA. In this phase we use the global search capability of MOGA to search for the initial parameter values of the neural network, thereby avoiding entrapment in local minima. In the second phase, the neural network set is trained to find a more optimal parameter set locally. The non-dominated parameter set obtained from the first phase is considered as the input

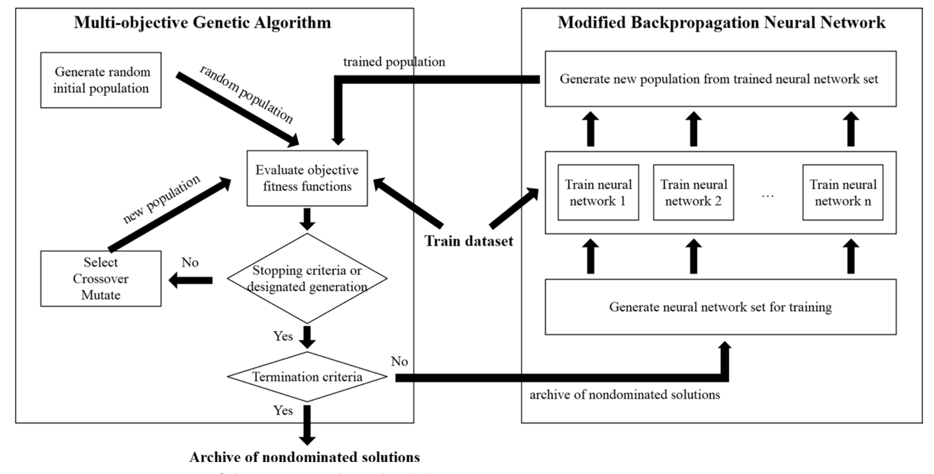
Multi-objective Genetic Algorithm
Generate random initial population
random population
trained population
Evaluate objective fitness functions
new population
Select Crossover Mutate
No
Stopping criteria or designated generation
Yes
Termination criteria
No
Yes
Archive of nondominated solutions
Train dataset
Modified Backpropagation Neural Network
Generate new population from trained neural network set
Train neural network 1
Train neural network 2
...
Train neural network n
Generate neural network set for training
archive of nondominated solutions

Fig. 1 Training session of the proposed methodology

archive. A neural network set is generated by selecting an excellent parameter set from the input archive, then back-propagation is used to update the weights for the neurons in order to bring the error function to a minimum. In this phase, we use the local search and fast learning capabilities of the neural network to refine the parameter set from the first phase, and we obtain a more optimal non-dominated parameter set. The non-dominated parameter set obtained in the second phase is used as the input of the first stage, and the training process of the two-phase algorithm is repeated until the termination criteria are met. In addition, a new self-adaptive parameter adjustment strategy is also used in the training process of the genetic algorithm and neural network.

The detailed implementation of the proposed algorithm is as follows:

Step 1: *Generate random initial population.* Create random initial population, and maintain it in a solution archive. The structure of the chromosomes that make up the entire population is illustrated in Fig. 2. The weight segment in chromosomes represents the weights between the input layer and hidden layer and also includes the weights between the hidden layer and output layer. The bias segment is dedicated to representing the biases of hidden nodes and output nodes. The gain factors of hidden nodes and output nodes are represented by the gain factor segment. After studying multiple training results of the neural network, the numerical range of neural network parameters is estimated, and the initial value of neural network parameters is limited between − 1000 and 1000.

Step 2: *Evaluate objective fitness functions.* Calculate the objective fitness function

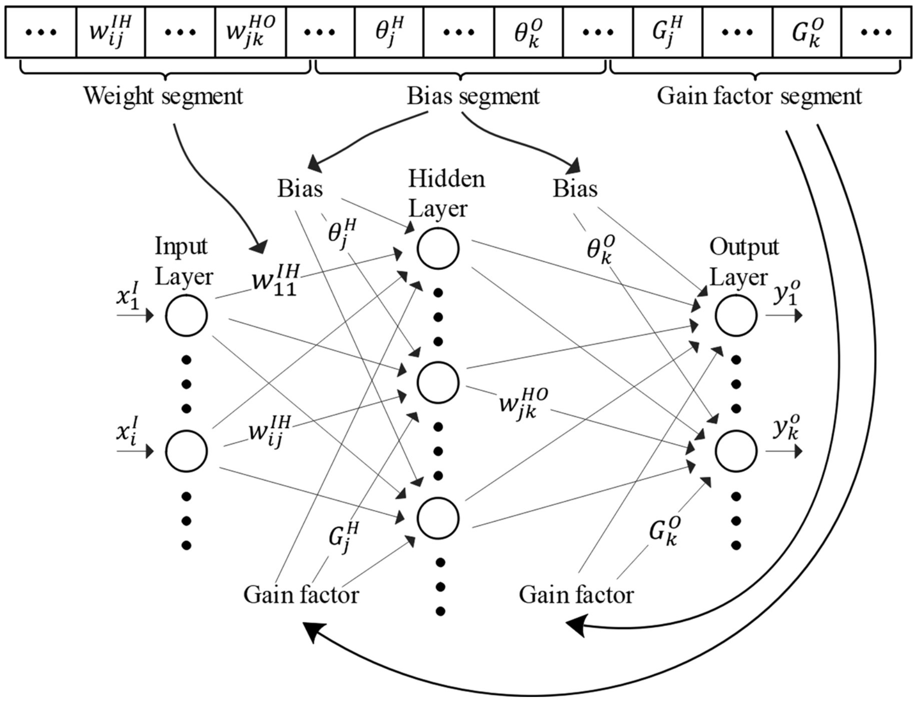

Fig. 2 The chromosome represents the neural network parameters

values for the neural network corresponding to a solution, then sort all solutions based on these values and update the solution archive. First, create neural networks based on the parameters represented by all chromosomes in the MOGA population. Then, these neural networks are used to classify the samples in the training dataset one by one. Next, calculate and record TPR and FPR for five output types. The TRP and FPR are the most important criteria for the classifier. At the same time, MSE is an important criterion for evaluating the performance of neural network classification. MSE is recorded and calculated as follows.

$$\mathrm{MSE} = \frac{1}{n}\sum_{l=1}^{n}(D_l - Y_l^o)^2 \tag{1}$$

where D_l and Y_l^o represent the desired output and actual output for the lth sample of the neural network, respectively, and n is the number of samples in the training dataset. After calculating the Avg TPR and Avg FPR for the types, they together with MSE constitute three objective fitness functions of the MOGA. Using Avg TRP instead of TRP as objective fitness function of the algorithm can avoid bias against specific attack types, especially those with less training samples such as R2L and U2R. For the same reason, Avg FRP is also used as an objective fitness function. Finally, all chromosomes are sorted and the solution archive is updated. The chromosomes are sorted according to the following two rules [40]:

(1) First, chromosomes are sorted according to non-inferior order, and the chromosomes with small non-inferior order values are ranked at the top of the solution archive.
(2) Second, chromosomes with the same non-inferior order are sorted according to crowded degree, where the less crowded chromosomes are closer to the top of the solution archive.

Among the above two rules, the first rule is made to find non-inferior solutions, and the second rule is set down to ensure that the distribution of non-inferior solutions is as dispersive as possible.

Step 3: *Stopping criteria or designated generation.* MOGA uses two different criteria to determine when to stop the solver. MOGA detects whether there is no change in the best fitness value for some number of generations (stall generation limit). MOGA also stops when the designated number of generations is reached: by default, this number is 300. If either of the two stopping criteria is reached, the algorithm will stop and go to step 5; otherwise, the algorithm will go to step 4.

Step 4: *Selection, crossover and mutation.* MOGA uses selection, crossover and mutation operators in the solution archives to generate new populations. The new population is added into the solution archive, then MOGA goes back to step 2 and the above steps are repeated. The MOGA used in this paper is a variant of Stud GA [41]. First, some of the best solutions in the archive obtained from step 2 are moved to a stallion archive, and the rest of the solutions are moved to the other temporary archive. Then, the linear ranking selection is used to select a solution from the stallion archive as a stallion and to select the other solution from the temporary archive. Next, the selected two solutions are randomly chosen for crossover to create two offspring by arithmetic crossover operator. Afterward, all the chromosomes resulting from the crossover operation are sent through a consequent mutation process. Finally, the selection, crossover and mutation operators are used repeatedly to generate new population until the maximum population size is reached.

Step 5: *Termination criteria.* The algorithm will terminate only if the following three conditions are met:

(1) The Avg TPR is greater than the designated value.
(2) The Avg FPR is smaller than the designated value.
(3) The number of non-inferior individual solutions is greater than the designated value.

If termination criteria are satisfied, the algorithm will terminate; otherwise, the algorithm will go to step 6.

Step 6: *Generate neural network set for training.* Calculate the objective fitness function values on the validation dataset for each solution in the archive, then sort all solutions based on their function values and select some of the best solutions to generate neural networks using the parameters represented by the solutions. The MBPNN [42] that is used in our approach adds gain factors G to change the steepness of the neural network activation function. During the learning process, gain factors change along with the weights and biases to speed up the convergence. The main modifications of the algorithm are as follows:

(1) The neural network activation function is still a sigmoid function, but the value range is changed to $[-0.5, +0.5]$. This can overcome the problem that the changes of weights and biases do not change the calculation when learning zero-value samples. The modified sigmoid is as follows:

$$f(x) = -0.5 + \frac{1}{1 + \mathrm{e}^{-x}} \tag{2}$$

(2) Let f be the neural network activation function and add gain factor G_j to the net input I_j, which is computed as the sum of the input features multiplied by the weights, so that the output y_j is defined as follows:

$$y_j = f(G_j I_j) \tag{3}$$

(3) The rule for gain factor updating is the same as the rule for weights and biases, and the updated value of the gain factor ΔG_j can be calculated as follows:

$$\Delta G_j = \eta \delta_j I_j / G_j \tag{4}$$

where η is learning rate and δ_j represents an error term at node j.

Step 7: *Neural network training.* Each created neural network from step 6 is trained on the training dataset by a back-propagation algorithm, which reduces the error values and update weights, biases and gain factors, so that the actual output values and desired output values become close enough. The method to update gain factors is expressed as Eq. (4).

Step 8: *Generation of new population from trained neural network set.* When the learning process for neural networks is completed, the objective fitness function values on the validation dataset are calculated for each trained neural network. Some of the best neural networks are selected and new chromosomes are constructed based on their parameters, and then, these chromosomes form a new population. The new population is added into the solution archive, then step 2 and the above steps are repeated.

2.1.2 The testing session

The testing session is depicted in Fig. 3. The testing session of the proposed approach integrates the predictions of several MBPNN classifiers to obtain the prediction of the final ensemble. The majority voting method is used to determine into which attack type a test sample is ultimately classified. The final attack type of a test sample is the one which receives the most votes.

The detailed implementation of the testing session is as follows:

Step 1: *Generation of a neural network set for testing.* Select some non-inferior solutions from the archive of non-dominated solutions obtained by genetic algorithm, and generate neural networks using the parameters represented by the solutions. The users can select several different solutions for combined

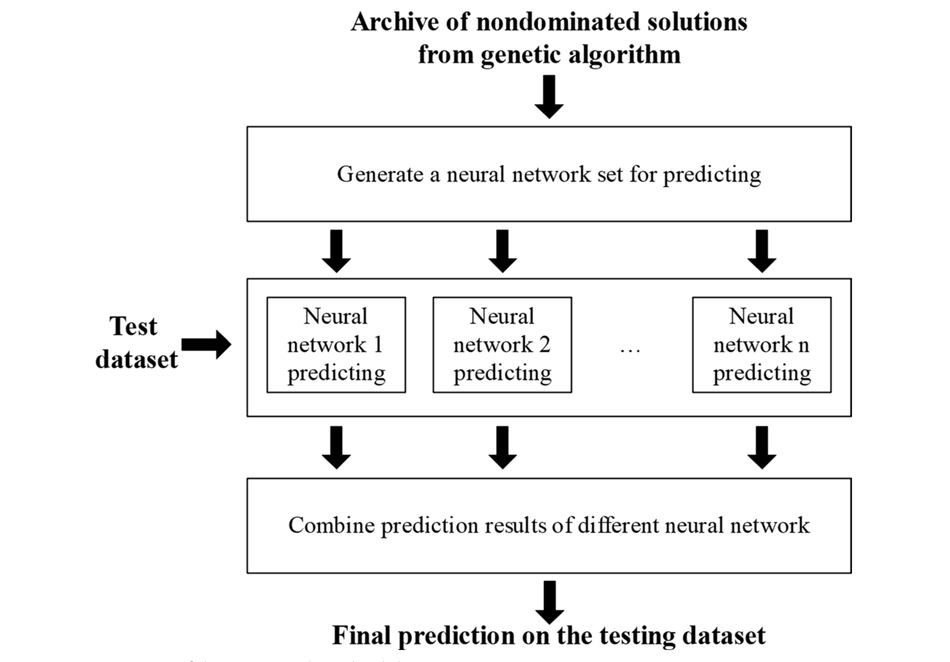
Archive of nondominated solutions from genetic algorithm
Generate a neural network set for predicting
Test dataset
Neural network 1 predicting
Neural network 2 predicting
…
Neural network n predicting
Combine prediction results of different neural network
Final prediction on the testing dataset

Fig. 3 Test session of the proposed methodology

prediction according to their own preferences. For example, they may choose a solution subset with large TPRs, a solution subset with small FPRs, or a combined solution subset with large TPRs and small FPRs.

Step 2: *Neural network prediction.* Each created neural network from step 1 is used to classify samples of the test dataset by the forward propagation algorithm of MBPNN, and the neural network releases a prediction result.

Step 3: *Combination of prediction results of different neural networks.* The prediction results from different MBPNN are combined by the majority voting method to yield the final output of the ensemble. For each individual sample, the prediction result of each MBPNN will receive one vote, and the final attack type of this sample is the one which receives the largest number of votes.

2.1.3 The combined classification method

In the testing session, some non-inferior solutions are selected from the archive of non-dominated solutions obtained in the training session, and an MBPNN set is generated for prediction. Therefore, how to choose these non-inferior solutions is critical to the performance of combined prediction.

A genetic algorithm is used to find the optimal combination solution set. Each chromosome represents one combination solution set, where the structure of the chromosomes is illustrated in Fig. 4. Each gene of the chromosomes corresponds to a solution in the solution archive. When the gene value is 1, it means that the corresponding solution is selected into the combination solution set, and when it is 0, this corresponding solution is not selected. Since the number of solutions in the solution archive is too large, we choose a subset of the archive to form chromosomes. The Avg TPR segment in chromosomes represents some solutions with maximum Avg TPRs selected from the solution archive. The AVG_FPR segment represents some solutions with minimum AVG_FPRs, and the accuracy segment represents some solutions with maximum accuracy values.

The GA takes Avg TPR, Avg FPR and accuracy of combination solution set on the training dataset as the optimization objectives. To simplify the calculation, the three objectives are combined into a single objective using a linear weighting method, as shown in the following formula:

$$maxF(l) = \sum_{k=1}^{3} a_k * f_k(I) \tag{5}$$

where $f_k(I)$ represents the kth objective, which is one of the three objectives (Avg TPR, Avg FPR and accuracy), and a_k is the weight of the kth objective.

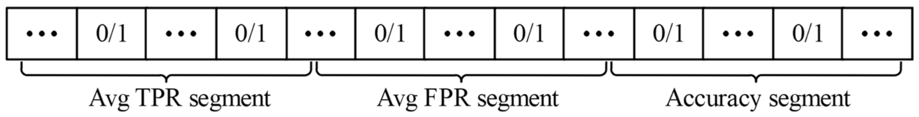

Fig. 4 The chromosome represents which solutions are selected in the combination solution set

2.2 The intrusion detection dataset

The performance of the proposed approach is measured based on the KDD cup 1999 [43] dataset, which is the most widely used dataset for validation of an IDS. Each record of the KDD dataset contains 41 feature attributes and 1 label attribute. The dataset contains five major types of attacks: normal, probe, denial of service (DoS), user-to-root (U2R) and remote-to-local (R2L). The last four are attack types, which can be subdivided into 39 different attack types. The dataset is very large, including 5 million training records and 2 million test records, so it is practically very difficult to use the whole dataset. In this study, we first remove records that have the same value for all features and then randomly select different records to form subsets that contain different proportions of normal and attack instances. The selected subsets used in our experiments are depicted in Table 1. Three data subsets are extracted: DATASUBSET1, DATASUBSET2 and DATASUBSET3. The number of samples in the three subsets increases successively: DATASUBSET1 extracts 12,250 records, DATASUBSET2 extracts 26,160 records and DATASUBSET3 extracts 145,586 records. With the increase of the number of samples, the classification features covered by the samples become more comprehensive, but the calculation cost is also significantly increased. By randomly extracting samples, each data subset is divided into three smaller subsets, namely training dataset, validation dataset and test dataset. The training dataset is used to fit the weights, biases and gain factors of MBPNN during the learning process. The validation dataset is used to compare the performance of different MBPNN classifiers and decide which ones to choose for the next step. The test dataset is a set of samples used only to assess the performance, and has never been exposed to a training session.

Data Transformation. We use one-hot encoding scheme to transform the three categorical features: protocol, service and state. A dimension is added for each new feature value found in the training set. The added dimensions indicate the presence or absence of a certain category, represented by binary [0, 1]. Following this scheme, the resulting datasets are extended into 123-dimensional features.

Data Normalization. We use min–max normalization to scale feature values, since the ranges of numerical features differ considerably from one another. We scale feature values linearly into the range of [−0.5, 0.5] by using the following equation.

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} - 0.5 \tag{6}$$

Table 1 Three data subsets of KDD cup 1999 dataset

Type	DATASUBSET1			DATASUBSET2			DATASUBSET3		
	Training	Validation	Test	Training	Validation	Test	Training	Validation	Test
Normal	1000	1000	1000	4000	2000	2000	70,266	8783	8783
Probe	1000	1000	1000	3000	1500	1500	1705	213	213
DoS	1000	1000	1000	4000	2000	2000	43,658	5457	5457
U2R	100	50	100	100	70	90	42	5	5
R2L	1000	1000	1000	2000	600	1300	799	100	100
Total	4100	4050	4100	13,100	6170	6890	116,470	14,558	14,558

where x' represents the normalized data, x represents the raw data, $\min(x)$ finds the minimum value for the current feature, and $\max(x)$ searches for the maximum value for the current feature.

Oversampling and Undersampling. The KDD dataset is a dataset with unbalanced numbers of samples, in which the number of normal and DoS samples is more than 70,000, and the number of U2R samples is less than 300. The classification of imbalanced datasets mostly benefits the majority type. This type of conundrum is resolved by oversampling or undersampling in order to generate the type-balanced dataset. In this paper, random undersampling is used to sample normal and DoS type, while Synthetic Minority Oversampling Technique (SMOTE) [44] is used to sample U2R and R2L type.

2.3 Experimental setup

To evaluate the proposed approach, an experimental application is implemented in C#. MBPNN is used as a basic classifier. The performance of the proposed technique is evaluated based on the benchmark KDD cup 1999 dataset. Avg TPR, Avg FPR and MSE are used as objective fitness functions of MOGA. The majority voting method is used to integrate the predictions of several basic classifiers to acquire the combined prediction for the final ensemble. The results of experiments are computed on a Windows PC with an AMD Ryzen 9 3900XT 12-Core, 3.80 GHz CPU and 32 GB RAM.

2.3.1 Performance metrics

In order to evaluate the performance of the proposed IDS, we use the following widely known metrics: accuracy, true positive rate and false positive rate, Avg TPR, Avg FPR, defined as follows:

(1) True positive (TP) is the number of attack records classified correctly;
(2) True negative (TN) is the number of normal records classified correctly;
(3) False positive (FP) is number of normal records classified incorrectly;
(4) False negative (FN) is the number of attack records classified incorrectly.

True positive rate (TPR), also known as detection rate, recall or sensitivity, is the proportion of positive cases that are correctly identified and is calculated as follows:

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{7}$$

False positive rate (FPR), also known as false alarm rate (FAR), is the proportion of positive cases that are incorrectly identified and is calculated as follows:

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}} \tag{8}$$

Accuracy is the proportion of the total number of predictions that is correct and is calculated as follows:

$$\text{Accuracy} = \frac{\text{TN} + \text{TP}}{\text{TN} + \text{TP} + \text{FN} + \text{FP}} \tag{9}$$

Average true positive rate (Avg TPR), also known as false alarm rate (Avg DR), is the average TPR of all major types and is calculated as follows:

$$\text{Avg TPR} = \frac{1}{K}\sum_{l=1}^{K}(\text{TPR}_l) \tag{10}$$

where TPR_l represents the TPR of the lth major types, and K is the number of major types in the dataset.

Average false positive rate (Avg FPR), also known as false alarm rate (Avg FAR), is the average FPR of all major types and is calculated as follows:

$$\text{Avg FPR} = \frac{1}{K}\sum_{l=1}^{K}(\text{FPR}_l) \tag{11}$$

The closer the values of these metrics (except for FPR and Avg FPR) are to one, the better the performance of IDS. The FPR and Avg FPR should be closer to zero for better performance.

2.3.2 Design of experiments

The proposed approach involves two algorithms: MOGA and MBPNN. These two algorithms require setting many hyperparameters, such as the hidden node number, maximum learning rate and maximum crossover probability. In this paper, through many iterations of experiments, the parameters are constantly adjusted, and the best parameter values are selected.

The implementation of MOGA requires some parameters as depicted in Table 2. Designated generation is used to set the maximum number of cycles for the genetic algorithm. In order to find more solutions, the population size is maintained at a large value. The stallion population is created to retain the few optimal solutions. Trained population size is the number of selected chromosomes, which is used to create MBPNN for phase 2. The probabilities of crossover and mutation change with the quality of the obtained solutions, and bad solutions will increase the probabilities but cannot exceed the set maximum values.

Table 2 Configuration of MOGA

Parameter	Values
Designated generation	400
Population size	300
Stallion population size	10
Trained population size	20
Maximum crossover probability	0.4
Maximum mutation probability	0.1

Table 3 Configuration of MBPNN

Parameter	Values
Designated training epochs number	100
Input nodes number	123
Hidden nodes number	3
Output nodes number	5
Maximum learning rate	0.5
Maximum initial value of neural network parameters	1000
Minimum initial value of neural network parameters	− 1000

Table 4 Configuration of GA

Parameter	Values
Number of MBPNN	45
Designated generation	300
Population size	30
Elite population size	3
Maximum crossover probability	0.4
Maximum mutation probability	0.1

The parameters of MBPNN are depicted in Table 3. The designated training epochs number is used to set the maximum number of cycles. The number of input nodes is the number of cyber intrusion feature attributes after one-hot encoding. The number of hidden nodes is kept small to reduce computation. The number of output nodes corresponds to five types of cyber intrusion. The learning rate increases when the network error is large: otherwise, it will decrease. The learning rate cannot exceed the set maximum value. Maximum and minimum initial values are used to set the range of the initial parameter values.

The parameters of GA for combined prediction are depicted in Table 4. The MBPNN number indicates how many solutions are selected from the solution archive to participate in the optimization calculation of the genetic algorithm. Designated generation is set to 300. The population size is set to 30, and the number of elite solutions is 3. The maximum crossover probability and mutation probability are set to 0.4 and 0.1, respectively.

3 Results and discussion

Here, the two-phase cycle training algorithm is executed to optimize parameter values for MBPNN. After that, we use GA to select some of the best solutions obtained by the training algorithm to create MBPNN classifiers for the final ensemble. Finally, each created MBPNN is used to classify samples on the test dataset, and the prediction results are combined by the majority voting method to yield the final output of the ensemble.

3.1 Results of different data subsets

The two-phase cycle training algorithm is applied to three data subsets as described in Table 1 and produces three sets of MBPNN-based ensemble solutions. We choose some

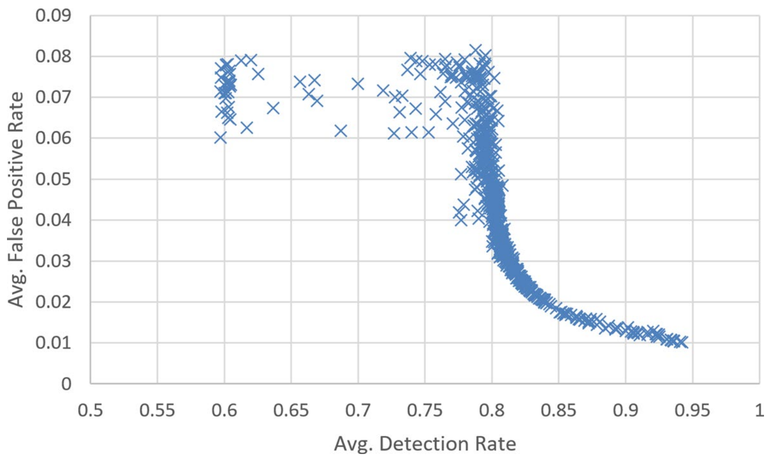

Fig. 5 Training performance of the proposed approach for DATASUBSET1

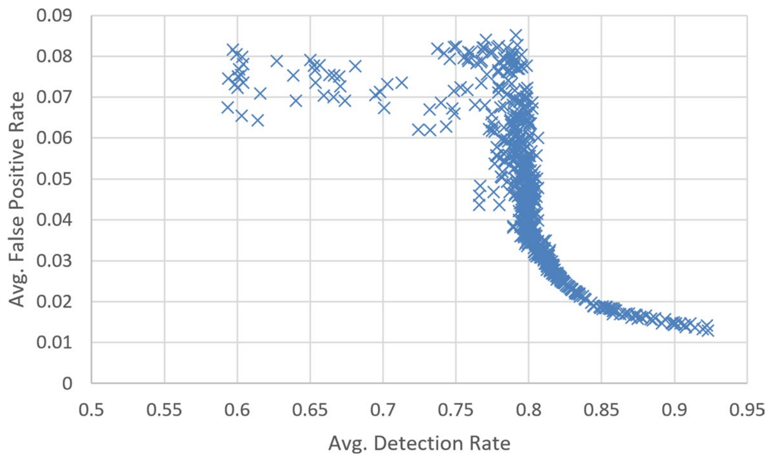

Fig. 6 Test performance of the proposed approach for DATASUBSET1

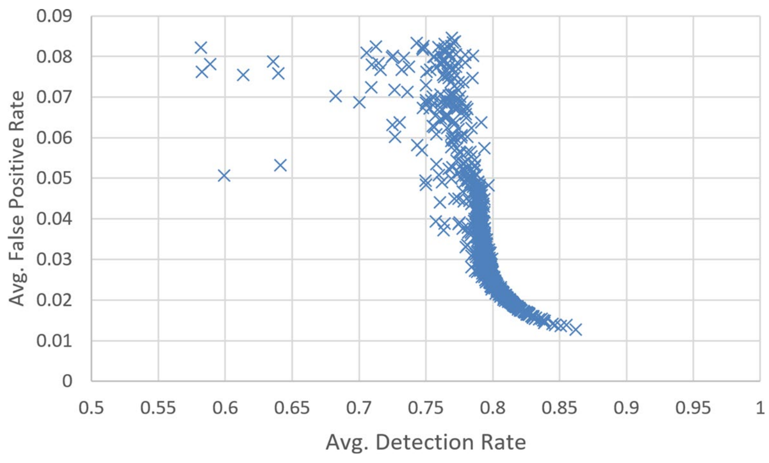

Fig. 7 Training performance of the proposed approach for DATASUBSET2

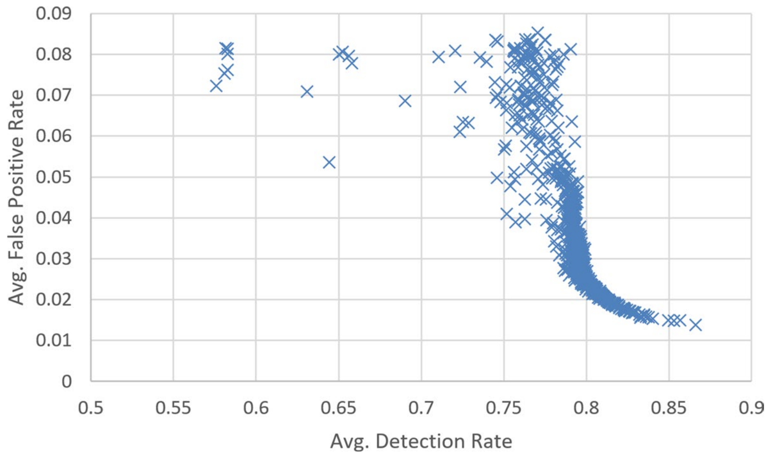

Fig. 8 Test performance of the proposed approach for DATASUBSET2

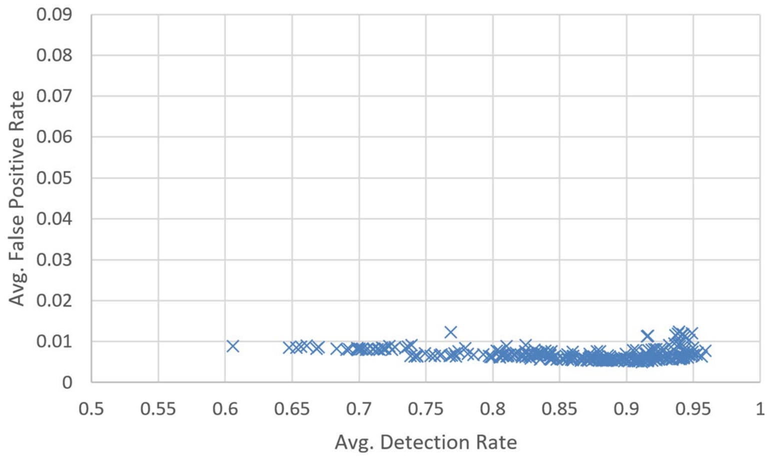

Fig. 9 Training performance of the proposed approach for DATASUBSET3

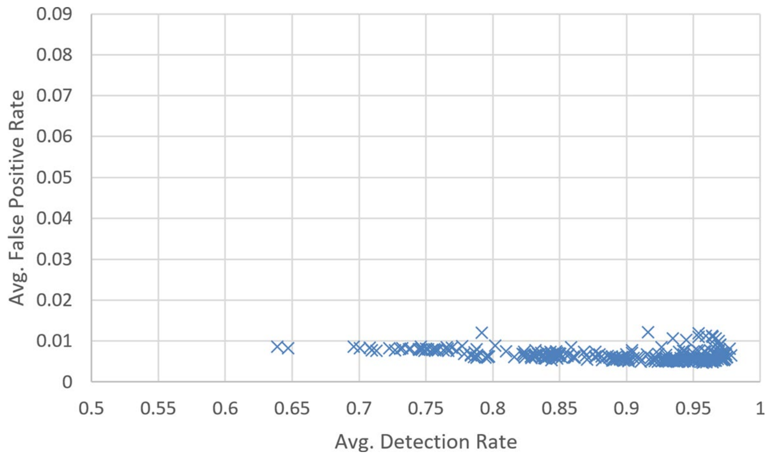

Fig. 10 Test performance of the proposed approach for DATASUBSET3

Table 5 Performance for different data subsets

Type	DATASUBSET1			DATASUBSET2			DATASUBSET3		
	BestAccu	BestDR	BestFAR	BestAccu	BestDR	BestFAR	BestAccu	BestDR	BestFAR
Normal									
DR	92.64	89.5	92.86	95.96	95.92	97.13	99.21	97.56	***99.28***
FAR	1.4	**0.93**	1.66	2.94	2.85	3.42	1.78	1.19	*1.89*
Probe									
DR	95.88	91.4	94.47	95.96	96.08	95.65	95.31	96.24	***96.71***
FAR	1.51	0.73	0.85	0.52	0.52	0.52	**0.18**	0.26	*0.24*
DoS									
DR	99.47	**100**	99.68	99.79	99.79	99.79	98.44	98.26	*98.3*
FAR	1.84	2.7	2.02	1.75	1.77	1.65	0.36	0.56	***0.19***
U2R									
DR	75.42	24.07	71.65	45.51	43.55	48.75	***100***	**100**	80
FAR	0.28	**0.08**	0.23	0.13	0.13	0.18	*0.11*	1.42	0.1
R2L									
DR	93.65	**97.77**	*95.11*	89.24	89.38	87.09	82	32	81
FAR	1.81	11.14	*2.15*	2.05	2.19	1.69	**0.03**	0.14	0.06

Table 6 Comparison of overall accuracy, detection rate and false alarm rate for different data subsets

	DATASUBSET1			DATASUBSET2			DATASUBSET3		
	BestAccu	BestDR	BestFAR	BestAccu	BestDR	BestFAR	BestAccu	BestDR	BestFAR
Accuracy	94.76	87.17	94.71	94.43	94.37	94.4	***98.75***	97.36	98.74
DR	98.55	**98.94**	98.29	96.96	97.05	96.47	*98.21*	98.78	98.11
FAR	15.23	33.88	14.89	11.38	11.73	10.52	*0.9*	3.56	**0.84**

of the best solutions to create MBPNN classifiers and then use these classifiers to predict the types of training samples and test samples. The performance of MBPNN classifiers on training data of DATASUBSET1 is depicted in Fig. 5, and the performance on test data is shown in Fig. 6. The performances of MBPNN classifiers on training and test data of DATASUBSET2 are depicted in Figs. 7 and 8, respectively, and the performances on training and test data of DATASUBSET3 are depicted in Figs. 9 and 10.

We choose one individual solution with the highest overall accuracy, one individual solution with the largest overall detection rate, and one individual solution with the smallest overall false alarm rate from the three data subsets to carry out prediction. The predicted performance for five types on the test dataset is shown in Table 5, and the overall performance is listed in Table 6. In Table 5, the highest DR and lowest FAR for each major type are emphasized in bold, and the italicized values represent the methods that have a good trade-off between DR and FAR. In Table 6, the best performance values are also emphasized in bold, and the italicized values are used to represent the best comprehensive performance.

From the results shown in these figures and tables, the following conclusions can be drawn:

Table 7 Performance of combined solutions

Attack type	BestAccu	BestDR	BestFAR	Combined
Normal				
DR	99.21	97.56	**99.28**	***99.28***
FAR	1.78	**1.19**	1.89	*1.7*
DoS				
DR	95.31	96.24	96.71	***98.36***
FAR	**0.18**	0.26	0.24	*0.19*
Probe				
DR	**98.44**	98.26	*98.3*	96.71
FAR	0.36	0.56	***0.19***	0.23
U2R				
DR	**100**	**100**	80	***100***
FAR	0.11	1.42	0.1	***0.09***
R2L				
DR	82	32	81	***86***
FAR	**0.03**	0.14	0.06	***0.03***

Table 8 Comparison of overall accuracy, detection rate and false alarm rate of combined solutions

Attack type	BestAccu	BestDR	BestFAR	Combined
Accuracy	98.75	97.36	98.74	**98*.81***
DR	98.21	**98.78**	98.11	*98.23*
FAR	0.9	3.56	0.84	***0.8***

(1) The Avg DR and Avg FPR of most solutions obtained by the algorithm are gathered together to form a clear Pareto front, which exhibits the excellent optimization performance of the algorithm.
(2) The smaller the number of samples in the data subset, the greater the number of divergent solutions that are not in the Pareto front, and the larger the range of the Pareto front. This shows that the algorithm can improve the detection performance by adding the number of samples.
(3) Although oversampling and undersampling are used, the algorithm has worse detection performance for types with small numbers of samples. The DRs of U2R on DATASUBSET1 and DATASUBSET2 are less than 76, while the DRs of other types are generally greater than 90. The same situation occurs for R2L of DATASUBSET3.
(4) As the number of samples in the data subset increases, the detection performance of the algorithm improves. Among performance metrics, the biggest improvement is in FAR. The FAR of DATASUBSET1 is approximately 15, the FAR of DATASUBSET1 is approximately 11, and the FAR of DATASUBSET3 is 1.

3.2 Results of combined experiment

An experiment called the combined experiment was performed, which classifies by using the majority voting method. A genetic algorithm is used to select some of the best solutions for the majority voting method. The results of the combined experiment are compared with the results of some excellent individual solutions. The predicted performances for five types of combined experiment are shown in Table 7, and the overall performance is listed in Table 8. In these tables, BestAccu, BestDR, and BestFAR, respectively, represent one solution with largest overall accuracy, with largest overall detection rate, and with smallest overall false alarm rate, which is the same as the previous description. The best performance values are also emphasized in bold, and the italicized values are used to represent the best comprehensive performance.

These tables show that although some individual solutions can achieve good performance, the combination of these solutions can still significantly improve their performance. The details of performance comparison are as follows:

(1) The detection performances of the combined classification method on normal, DoS, U2R and R2L are better than all individual solutions. Only the performance on probe is worse than the performance of the BestFAR individual solution.
(2) The overall accuracy and FAR of the combined classification method are the best among all the solutions, and the DR of the combined classification method is also very close to the optimal value.
(3) The comprehensive performance of the combined method is significantly better than all individual solutions. This also proves that the combined classification method is a feasible classification method with better performance for intrusion detection.

3.3 Discussion

More experiments are conducted to validate the performance of the proposed approach. The experimental results of TPC-MOGA-MBPNN are compared with the results of some well-known models reported in the literature, as shown in Table 9. It is evident

Table 9 Comparison with other methods

Method	Normal	DoS	Probe	U2R	R2L	Accuracy	DR	FAR
Multi-level SVM-ELM [45]	98.13	99.54	87.22	21.93	31.39	95.75	95.17	1.87
NFC[36]	98.2	99.5	84.1	14.1	31.5	N/A	95.2	1.9
Genetical algorithm [45]	69.5	99.4	71.1	18.9	5.4	90	94.95	30.46
SVM+BIRCH clustering [45]	99.3	99.5	97.5	19.7	28.8	95.7	N/A	0.7
MOGFIDS [45]	98.36	97.2	88.6	15.79	11.01	93.2	91.96	1.6
Association rules [45]	99.47	96.6	74.8	3.8	1.21	92.4	N/A	0.53
Multiclass SVM [45]	99.6	96.8	75	5.3	4.2	92.46	90.74	**0.43**
Winning the KDD99 [45]	99.5	97.1	83.3	13.2	8.4	93.3	91.81	0.55
t-SNE SVM [46]	**99.85**	99.62	**98.89**	86.8	96.51	N/A	N/A	N/A
Hybrid classifier [47]	96.8	98.66	93.4	46.97	71.43	96.77	96.77	3
DT-SVM [48]	99.7	**99.92**	98.57	48	37.8	N/A	N/A	N/A
GA FPM [49]	N/A	98.85	98.05	78.7	**98.7**	N/A	N/A	N/A
Proposed approach (combined)	99.28	98.36	96.71	**100**	86	**98.81**	**98.23**	0.8

that the proposed TPC-MOGA-MBPNN outperforms the cited approaches with respect to consistently good overall accuracy and DR.

4 Conclusion

In this paper, a novel TPC-MOGA-MBPNN algorithm based on MOGA and MBPNN is proposed for effective intrusion detection. The proposed approach is capable of producing a pool of non-inferior individual solutions that exhibit classification trade-offs for the user. By using certain heuristics or prior domain knowledge, a user can select an ideal solution or combined solution as per application specific requirements. The proposed approach attempts to tackle the issues of low DR, high FPR and lack of classification trade-offs in the field of intrusion detection. The proposed approach consists of encoding of chromosomes that provide optimized parameter values of the MBPNN. MOGA is employed to build a multi-objective optimization model that generates Pareto optimal solutions with simultaneous consideration of Avg TPR, Avg FPR and MSE in the dataset. A two-phase cycle training algorithm-based approach can rapidly generate numerous non-inferior solutions. In the first phase, a MOGA tries to find the Pareto optimal parameter set for the neural networks. In the next phase, some selected MBPNNs based on chromosomes obtained by MOGA are trained to find a more optimal parameter set locally. The non-dominated parameter set obtained in the second phase is used as the input of the first stage, and the training of the two-phase algorithm is repeated until the termination criteria are reached.

The KDD cup 1999 dataset is used to demonstrate and validate the performance of the proposed approach for intrusion detection. The proposed approach exhibits the excellent optimization performance of the algorithm and a very clear Pareto front has been obtained. The optimized set of MBPNN exhibits the classification trade-offs for the users. The user may select an ideal solution as per application-specific requirements. We also demonstrate that combining a few MBPNN classifiers represents a feasible classification method and yields better performance than using an individual MBPNN for classification. A genetic algorithm is used to find the optimal MBPNN combination and can discover an optimized set of MBPNN with good accuracy and detection rate from benchmark datasets. The result shows that the proposed approach could achieve an accuracy of 98.81% and a detection rate of 98.23%, which outperform most systems of previous works found in the literature. The results of this work have also provided an alternative with respect to the issue of selecting an optimal solution among the non-dominated Pareto optimal solutions.

The main challenges of the proposed approach are as follows:

(1) The MOGA requires substantial time to compute fitness functions in various generations. This may be overcome by computing the function values in parallel, or limiting the population size.
(2) The performance of the proposed approach is affected by the number of samples. A large number of samples are needed for excellent performance, which inevitably requires considerable calculation.
(3) The computing power of edge computing devices was usually weak, but the proposed algorithm requires strong computing power. This problem can be solved by

training the model on a high-performance PC, and only employing the edge nodes to run the trained model for prediction.

(4) The proposed approach only uses a small subset of the benchmark dataset for validation, and its applicability can be validated by more experiments on real cyber traffic in the field of intrusion detection.

Abbreviations

GA: Genetic algorithm; MOGA: Multi-objective genetic algorithm; MBPNN: Modified back-propagation neural network; Avg FPR: Average false positive rate; MSE: Mean squared error; Avg TPR: Average true positive rate; NIDS: Network-based intrusion detection system; ANNs: Artificial neural networks; ANN: Artificial neural network; DDoS/DoS: Distributed denial of service; RNN-IDS: Intrusion detection using recurrent neural networks; DCNN: Deep convolutional neural network; IDS: Intrusion detection system; ENN: Evolutionary neural network; EA: Evolutionary algorithm; ANNIDS: Artificial neural network-based intrusion detection system; MLP: Multilayer perceptron; DIS: Direct acyclic graph information solicitation; MOO: Multi-objective optimization; NSGA II: Non-dominated sorting genetic algorithm; AMGA2: Micro Genetic Algorithm2; U2R: User-to-root; R2L: Remote-to-local; TP: True positive; TN: True negative; FP: False positive; FN: False negative; TPR: True positive rate; FPR: False positive rate; FAR: False alarm rate.

Acknowledgements

This work was supported by the National Key Research and Development Program of China (Grant No. 2018YFC1405700) and the Industry University Research Cooperation Project of Jiangsu Province (Grant No. BY2019005).

Authors' contributions

Yiguang Gong, Yunping Liu, and Chuanyang Yin conceptualized and designed the review. Yiguang Gong wrote the paper. All authors reviewed and edited the manuscript. All authors read and approved the final manuscript.

Availability of data and materials

The KDD cup 1999 dataset is used to demonstrate and validate the performance of the proposed approach for intrusion detection. The dataset can be accessed from the following website: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

Declarations

Competing interests

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

Received: 29 October 2020 Accepted: 10 June 2021
Published online: 10 July 2021

References

1. W. Zhong, X. Yin, X. Zhang, S. Li, W. Dou, R. Wang, L. Qi, Multi-dimensional quality-driven service recommendation with privacy-preservation in mobile edge environment. Comput. Commun. **157**,116–123 (2020). https://doi.org/10.1016/j.comcom.2020.04.018
2. L. Qi, C. Hu, X. Zhang, M.R. Khosravi, S. Sharma, S. Pang, T. Wang, Privacy-aware data fusion and prediction with spatial-temporal context for smart city industrial environment. IEEE Trans. Ind. Inform. **17**(6), 4159–4167 (2020). https://doi.org/10.1109/TII.2020.3012157
3. X. Xiaolong, Q. Huang, Y. Zhang, S. Li, L. Qi, W. Dou, An lsh-based offloading method for iomt services in integrated cloud-edge environment. ACM Trans. Multimed. Comput. Commun. Appl. (TOMM) **16**(3s), 1–19 (2021). https://doi.org/10.1145/3408319
4. X. Xu, H. Qihe, X. Yin, M. Abbasi, M.R. Khosravi, L. Qi, Intelligent offloading for collaborative smart city services in edge computing. IEEE Internet Things J. **7**(9), 7919–7927 (2020). https://doi.org/10.1109/JIOT.2020.3000871
5. Q. Liu, Y. Tian, W. Jie, T. Peng, G. Wang, Enabling verifiable and dynamic ranked search over outsourced data. IEEE Trans. Serv. Comput (2019). https://doi.org/10.1109/TSC.2019.2922177
6. Z. Cai, X. Zheng, A private and efficient mechanism for data uploading in smart cyber-physical systems. IEEE Trans. Netw. Sci. Eng. **7**(2), 766–775 (2018). https://doi.org/10.1109/TNSE.2018.2830307
7. L. Qi, X. Wang, X. Xu, W. Dou, S. Li, Privacy-aware cross-platform service recommendation based on enhanced locality-sensitive hashing. IEEE Trans. Netw. Sci. Eng (2020). https://doi.org/10.1109/TNSE.2020.2969489
8. L. Wang, X. Zhang, T. Wang, S. Wan, G. Srivastava, S. Pang, L. Qi, Diversified and scalable service recommendation with accuracy guarantee. IEEE Trans. Comput. Soc. Syst (2020). https://doi.org/10.1109/TCSS.2020.3007812
9. L. Wang, X. Zhang, R. Wang, C. Yan, H. Kou, L. Qi, Diversified service recommendation with high accuracy and efficiency. Knowl.-Based Syst. **204**, 106196 (2020). https://doi.org/10.1016/j.knosys.2020.106196
10. R. Heady, G. Luger, A. Maccabe, M. Servilla. The architecture of a network level intrusion detection system, p. 8 (1990). https://doi.org/10.2172/425295

11. Melissa Michael, The state of cyber security (2017). https://blog.f-secure.com/the-state-of-cyber-security-2017
12. I. Manzoor, N. Kumar, A feature reduced intrusion detection system using ANN classifier. Expert Syst. Appl. **88**, 249–257 (2017)
13. R. Vijayanand, D. Devaraj, B. Kannapiran, Intrusion detection system for wireless mesh network using multiple support vector machine classifiers with genetic-algorithm-based feature selection. Comput. Secur. **77**, 304–314 (2018)
14. L. Li, Y. Yu, S. Bai, Y. Hou, X. Chen, An effective two-step intrusion detection approach based on binary classification and k-nn. IEEE Access **6**, 12060–12073 (2018)
15. N. Farnaaz, M.A. Jabbar, Random forest modeling for network intrusion detection system. Procedia Comput. Sci. **89**, 213–217 (2016)
16. R. Vinayakumar, M. Alazab, K.P. Soman, P. Poornachandran, A. Al-Nemrat, S. Venkatraman, Deep learning approach for intelligent intrusion detection system. IEEE Access **7**, 41525–41550 (2019)
17. A. Cemerlic, L. Yang, J.M. Kizza, Network intrusion detection based on Bayesian networks, in SEKE, pp. 791–794 (2008)
18. Z. Cataltepe, U. Ekmekci, T. Cataltepe, I. Kelebek, Online feature selected semi-supervised decision trees for network intrusion detection, in *NOMS 2016—2016 IEEE/IFIP Network Operations and Management Symposium*, pp. 1085–1088 (2016)
19. E. Hodo, X. Bellekens, A. Hamilton, P.-L. Dubouilh, E. Iorkyase, C. Tachtatzis, R. Atkinson, Threat analysis of iot networks using artificial neural network intrusion detection system, pp. 1–6 (2016)
20. C. Yin, Y. Zhu, J. Fei, X. He, A deep learning approach for intrusion detection using recurrent neural networks. IEEE Access **5**, 21954–21961 (2017)
21. S. Naseer, Y. Saleem, S. Khalid, M.K. Bashir, J. Han, M.M. Iqbal, K. Han, Enhanced network anomaly detection based on deep neural networks. IEEE Access **6**, 48231–48246 (2018)
22. I. Benmessahel, K. Xie, M. Chellal, A new evolutionary neural networks based on intrusion detection systems using multiverse optimization. Appl. Intell. **48**(8), 2315–2327 (2018)
23. A.A. Anitha, L. Arockiam, Annids: artificial neural network based intrusion detection system for internet of things. Int J Innovative Technol Exploring Eng (IJITEE) **8**(11), 2583–2588 (2019)
24. Z. Sun, P. Lyu, Network attack detection based on neural network LSTM, in *2019 2nd International Conference on Mechanical, Electronic and Engineering Technology*, pp.12–17 (2019)
25. A. Shenfield, D. Day, A. Ayesh, Intelligent intrusion detection systems using artificial neural networks. ICT Express **4**(2), 95–99 (2018)
26. N. Talhar, Effective denial of service attack detection using artificial neural network for wired lan, in *2016 International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES)*, pp. 229–234. IEEE (2016)
27. M. Paliwal, U.A. Kumar, Neural networks and statistical techniques: a review of applications. Expert Syst. Appl. **36**, 2–17 (2009)
28. F. Ahmad, N.A.M. Isa, Z. Hussain, S.N. Sulaiman, A genetic algorithm-based multi-objective optimization of an artificial neural network classifier for breast cancer diagnosis. Neural Comput. Appl. **23**(5), 1427–1435 (2013)
29. X.-Y. Cao, H.-L. Yu, Y.-Y. Zou, Character recognition based on genetic algorithm and neural network, in *Proceedings of the 2012 International Conference on Information Technology and Software Engineering*, pp. 915–923. Springer (2013)
30. G. Kumar, K. Kumar, A multi-objective genetic algorithm based approach for effective intrusion detection using neural networks, in *Intelligent Methods for Cyber Warfare*, pp. 173–200. Springer (2015)
31. H.A. Abbass, Pareto neuro-evolution: constructing ensemble of neural networks using multi-objective optimization, in *The 2003 Congress on Evolutionary Computation, 2003. CEC'03*, vol. 3, pp. 2074–2080. IEEE (2003)
32. C.M. Fonseca, P.J. Fleming, et al, Genetic algorithms for multiobjective optimization: formulation discussion and generalization, in *Icga*, vol. 93, pp. 416–423. Citeseer (1993)
33. K. Deb, A. Pratap, S. Agarwal, T.A.M.T. Meyarivan, A fast and elitist multiobjective genetic algorithm: Nsga-ii. IEEE Trans. Evolut. Comput. **6**(2), 182–197 (2002)
34. X. Xiaolong, B. Shen, X. Yin, M.R. Khosravi, S. Wan, Edge server quantification and placement for offloading social media services in industrial cognitive iov. IEEE Trans. Ind. Inform. **17**(4), 2910–2918 (2021)
35. S. Elhag, A. Fernández, A. Altalhi, S. Alshomrani, F. Herrera, A multi-objective evolutionary fuzzy system to obtain a broad and accurate set of solutions in intrusion detection systems. Soft Comput. **23**(4), 1321–1336 (2019)
36. M. Stehlik, A. Saleh, A. Stetsko, V. Matyas, Multi-objective optimization of intrusion detection systems for wireless sensor networks, in *Artificial Life Conference Proceedings 13*, pp. 569–576. MIT Press (2013)
37. X. Xiaolong, X. Liu, X. Zhanyang, F. Dai, X. Zhang, L. Qi, Trust-oriented iot service placement for smart cities in edge computing. IEEE Internet Things J. **7**(5), 4084–4091 (2020)
38. S. Tiwari, G. Fadel, K. Deb, Amga2: improving the performance of the archive-based micro-genetic algorithm for multi-objective optimization. Eng. Optim. **43**(4), 377–401 (2011)
39. Y. Gong, Y. Liu, C. Yin, Z. Fan, A two-phase cycle algorithm based on multi-objective genetic algorithm and modified bp neural network for effective cyber intrusion detection, in *International Conference on Machine Learning for Cyber Security*, pp. 73–88. Springer (2020)
40. F. Ye, L. Nannan et al., Multiobjective optimization method based on pareto solution and its application. Lift. Transp. Mach. **2006**(9), 13–15 (2006)
41. W. Khatib, P.J. Fleming, The stud ga: a mini revolution? In *International Conference on Parallel Problem Solving from Nature*, pp. 683–691. Springer (1998)
42. Zhu J, Non-classical mathematical methods for intelligent systems (2001)
43. KDD Cup, The UCI KDD Archive. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html (1999)
44. N.V. Chawla, K.W. Bowyer, L.O. Hall, W.P. Kegelmeyer, Smote: synthetic minority over-sampling technique. J. Artif. Intell. Res. **16**, 321–357 (2002)
45. W.L. Al-Yaseen, Z.A. Othman, M.Z.A. Nazri, Multi-level hybrid support vector machine and extreme learning machine based on modified k-means for intrusion detection system. Expert Syst. Appl. **67**, 296–303 (2017)
46. Y. Hamid, M. Sugumaran, A t-sne based non linear dimension reduction for network intrusion detection. Int. J. Inf. Technol. **12**(1), 125–134 (2020)

47. C. Xiang, P.C. Yong, L.S. Meng, Design of multiple-level hybrid classifier for intrusion detection system using Bayesian clustering and decision trees. Pattern Recognit. Lett. **29**(7), 918–924 (2008)
48. S. Peddabachigari, A. Abraham, C. Grosan, J. Thomas, Modeling intrusion detection system using hybrid intelligent systems. J. Netw. Comput. Appl. **30**(1), 114–132 (2007)
49. P.U. Kadam, M. Deshmukh, Real-time intrusion detection with genetic, fuzzy, pattern matching algorithm, in *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, pp. 753–758. IEEE (2016)

Publisher's Note