Pak. J. Statist.
2018 Vol. 34(1), 01-14

NEW APPROACH TO IMPROVE ANOMALY DETECTION USING A NEURAL NETWORK OPTIMIZED BY HYBRID ABC AND PSO ALGORITHMS

Waheed Ali H.M. Ghanem[1,2] and **Aman Jantan**[1]
[1] School of Computer Science, Universiti Sains Malaysia
Email: waheed.ghanem@gmail.com
[2] Faculty of Education-Saber, Faculty of Engineering,
University of Aden, Aden, Yemen. Email: aman@cs.usm.my

ABSTRACT

Intrusion detection is one of the most significant concerns in network safety. Improvements have been proposed from various perspectives. One such proposal is improving the classification of packets in a network to determine whether the classified packets contain harmful packages. In this study, we present a new approach for intrusion detection using a hybrid of the artificial bee colony (ABC) algorithm and particle swarm optimization (PSO) to develop an artificial neural network with increased precision in classifying malicious from harmless traffic in a network. Our proposed algorithm is compared with four other algorithms employed in WEKA tool, namely, radial basis function, voted perceptron, logistic regression, and multilayer perceptron. The system is first prepared, and then suitable biases and weights, for the feed-forward neural network are selected using the hybrid ABC–PSO algorithm. Then, the network is retrained using the prepared information, which has been generated from the ideal weights and biases, to establish the intrusion-detection model. The KDD Cup 1999 data set is used as the information set in comparing our proposed algorithm with the other algorithms. The experiment shows that the proposed method outperforms the four classification algorithms and is suitable for network intrusion detection.

KEYWORDS

Intrusion Detection System; Data Mining; Feed-forward Neural Network; Artificial Bee Colony Algorithm; Particle Swarm Optimization.

1. INTRODUCTION

Incidents of system assaults are currently growing in number, which inevitably increases request for intrusion detection systems (IDSs). These systems were first created in 1980 by Anderson [1] and were perfected by Denning in 1987 [2]. An IDS can be a device or a software tool. It has been constantly improved since its introduction in the 1980s. The main purpose of an IDS is to detect and react to intrusive and harmful activities that target system resources and facilities. This objective can be achieved by paying close attention to the activities of a system and analyzing its networks [3]. IDSs can be categorized according to anomalies and misuse [4]. Anomalies and misuse are detected by different means. To detect misuse, an IDS searches for the attack fingerprint in a large database in which all attack signatures are stored. By contrast, an IDS perceives how a system behaves differently from its usual behavior to detect anomalies.

Recently, both data mining and neural networks (NNs) have been playing crucial roles in improving the performance quality of IDSs. They facilitate the process of categorizing the kinds of attacks to calculate the effectiveness of an IDS. The main purpose of a data mining procedure is to obtain simulated information from big knowledge warehouses and then convert it into a data structure that can be understood [5]. Artificial NNs (ANNs) are considered the most important parts of computer intelligence. They are categorized into supervised learning NNs or unsupervised learning NNs; as their names suggest, the former has to learn under the supervision of a person, whereas the latter does not require supervision [6]. To be successful in its activity, an IDS depends on the following factors [7]: its architecture, the training algorithm, and the features utilized during training. These factors contribute to the difficulty in achieving an optimal IDS design [8]. Reference [9] employed heuristic algorithm based methods to obtain a good ANN module. ANNs have been created in different kinds of multilayer NNs. The majority of the applications utilize feed-forward NNs (FFNNs) that apply the typical back propagation (BP) learning method. This type of training is typically completed by utilizing the BP gradient descent method [10]. Given that this algorithm is gradient based, some issues may be encountered when using it. One of the most notable problems is getting stuck in a local minimum as a result of the slightly low speed of convergence [10]. Global optimum search methods can avoid local minima and are frequently utilized to regulate the weight of a multilayer perceptron (MLP). Examples of these methods include artificial bee colony (ABC) [7, 11], particle swarm optimization (PSO) [12, 13], Bat algorithm [14, 15], evolutionary algorithms (EAs), simulated annealing (SA), and ant colony optimization (ACO). These methods can also be used to eliminate problems in standard and typical algorithms.

2. FEEDFORWARD ARTIFICIAL NEURAL NETWORK

An ANN comprises a group of interconnected processing units (Figure 1), which are also known as fake or mock neurons or nodes [4, 16]. Equation (1) can describe the output of the i^{th} artificial neuron (AN).

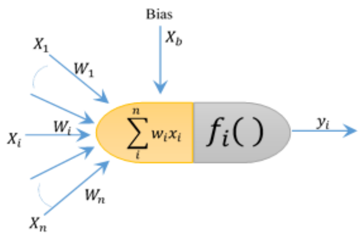

Figure 1: The Processing Unit (Neuron)

$$y_i = f_i\left(\sum_{i=1}^{n} w_{ij} x_i + \theta_i\right) \quad (1)$$

Each artificial neuron can receive inputs, which are also known as signals. These signals can come from either the environment or from other artificial neurons. Each signal has several characteristics: x_i has a w_{ij} weight that can either strengthen or deplete the input signal. An artificial neuron uses a function f_i to activate the output signal y_i after it has computed the input signal. As evident from the previous explanation, the y_i in the output signal of the artificial node x_i is the input signal of the node as well as the weight that can either strengthen the signal or deplete it. $\boldsymbol{\theta_i}$ can be defined as the bias or the threshold of an artificial neuron, and function f_i is responsible for activating the output signal of a node. The activation function is generally defined by a nonlinear function, such as the Heaviside function or a sigmoid function, but also by a Gaussian function. Several other types of functions can perform well in this situation.

Two-layer FFNNs are the most popular NNs used in practical applications, for example, in approximating functions [17, 18, 19]. Two-layer FFNNs are also suitable for categorizing nonlinearly divisible patterns [20, 21]. Experiments have proven that two-layer FFNNs can approximate nearly any continuous or discontinuous function [20].

In this study, a two-layer FFNN with one input layer, one output layer, and one hidden layer is proposed. n indicates the number of input neurons, h denotes the number of hidden neurons, and m signifies the number of output nodes. When utilizing FFNNs, the following set of key tasks should be focused on. First, a combination of weights and biases that provides the minimum error for such system should be obtained to improve the method in which it is employed. Second, a suitable structure for the FFNN should be established to achieve the final goal. The last task is to utilize an unconventional algorithm to regulate the parameters of the gradient-based learning method [19]. Based on [19, 22], the convergence of a BP method depends on the initial values of the parameters**.**

3. ARTIFICIAL BEE COLONY ALGORITHM

In order to optimize numerical issues, KARABOGA put forward in 2005 the Artificial Bee Colony (ABC) method [23]. Quite a few developments were made by the author of this method alongside some researchers who showed interest in the matter [24, 25, 26]. The ABC had as an inspiration and starting point the behavior of bee swarms, which was found to be rather intelligent and also foraging. It permitted the optimization algorithm to have a very simple, stochastic, but nonetheless robust form, based on the population of bees. The bees were categorized in to three types; employed, onlookers and scouts.

There are many tasks performed by a bee in the colony, and the most important of these tasks is to find out locations of food sources. Having obtained this information, it is evaluated through the consideration of the quality of food [23]. ABC algorithm depends on two criteria: the solution and the quality of the solution. A possible solution represents the location of the food source. The quality of the solution is comparable to the amount of nectar in the source location. The fitness is determined by the quality of nectar in the food source and its amount. The bees in colony classify into working bees (employed bees) and non-working bees (unemployed bees). The unemployed bees can as well be classified into onlooker bees and scout bees. The ABC Algorithm contains repetitive steps. Anyhow, there are two particular steps that are vital to the evolution of the ABC population. The first step refers to the process of finding new food location within a certain area, whereas the second step refers to the process of picking up the best food locations by comparing them to previous experiences. Also, in order to accomplish these two steps the bees have to follow a four step workflow: the first step is instauration, the second relies on employed bees, the third relies on onlooker bees and the fourth relies on the scout bees. These four steps are detailed below.

3.1 Initialization Phase and *Optimization* Problem Parameters

At first, the ABC algorithm suggests a proportionally distributed population that has solution number (SN) solutions. Among the SN solutions, each solution x_i (i = 1, 2... SN) represents a D-dimensional vector, where D is the number of variations in the optimization of the coordinates of the problem and x_i refers to the *i*th food location for the bee population. The food location can be determined by the following equation.

$$x_i^j = x_{min}^j + r\ and\ (0{,}1)\left(x_{max}^j + x_{min}^j\right) \tag{2}$$

In Equation (2), $x_{max}^j\ and\ x_{min}^j$ are the bounds of x_i in the *j*th dimension. Moreover, the ABC algorithm relies on three coordinates of control. First, the number of food sources depends on the population of bees. Second, the maximum cycle number can determine the maximum number of generations. Finally, a limit is used to determine the number of accepted generations after which the food sources that have not been improved will be removed from the locations used by that particular bee population. Once a food source is found and given to employed bees, the correct option that is specific for optimizing the coordinates of the problem is executed, and the bees reach the final result, in which Equation (3) is used to obtain every fitness value offered by each food location.

$$fit_i(t) = \begin{cases} \frac{1}{1+f_i(t)}\ if\ (f_i(t) \geq 0) \\ 1 + abc\left(f_i(t)\right)\ otherwise \end{cases} \tag{1}$$

In Equation (3), fit$_i$(*t*) represents the fitness value of the *i*th food location. The result is obtained by using food locations. An option is correct when it specifically optimizes the coordinates of the problem.

3.2 Employed Bee Phase

In this phase, the employed bees use the information obtained from personal experiences and the quantity of nectar of the solution to implement changes on the actual solution. In case the quantity of nectar in the newly found location is greater than that in the previous food locations, the bee population starts procuring food from the new location and abandons the last one [27, 28]. The following equation determines how a food source is modified:

$$v_i^j = x_i^j + \emptyset_{ij}\left(x_i^j + x_k^j\right) \tag{4}$$

where $\emptyset_{ij}\left(x_i^j + x_k^j\right)$ represents the length of the step, and $k \in$ {1, 2 ... SN} and $j \in$ {1, 2 ... *D*} are two indices that are selected at random. *k* is not equal to *i* because the length of the step has an important contribution, and $\emptyset_{ij}$ is only a number within the range of [0, 1].

3.4 Onlooker Bee Phase

The onlooker bee phase can be initiated only after the employed bee phase. In this phase, the quantity of nectar that has been collected during the employed bee phase is communicated. In communicating this information, the bees have to include the updated solution of their food location and the coordinates of this location. Onlooker bees also examine the information they hold and choose an option with probability $\mathcal{P}_i$.

$$p_i = \frac{fit_i}{\sum_{i=1}^{sn} fit_i} \tag{5}$$

In Equation (5), fit_{i} is the quantity of nectar that is determined by the i^{th} solution. Similar to employed bees, an onlooker bee updates the location of the new food source in its memory and compares the nectar of this new source to the nectar of another source. In case the quantity of nectar is greater than the one provided by the last food source, the bees remember the new food source and forget about the unproductive source.

3.5 Scout Bee Phase

In this phase, an employed bee transforms into a scout bee in case the employed bee makes a connection with the unproductive food source. Moreover, the food location is replaced with another food location that can be found within the search area. The scout bee phase can be initiated once the place where the food source is located has not been updated for numerous cycles, which causes the bees to think that that particular food source has been left behind. In the ABC algorithm, the number of cycles is an important control indicator and is called the limit for abandonment. In case the abandoned food source is x_i, scout bees replace the food source with another x_i in the following manner:

$$x_i^j = x_{min}^j + rand(0,1)\left(x_{max}^j + x_{min}^j\right) \forall j = 1,2 \dots D \tag{6}$$

In equation (6) the $x_{max}^j + x_{min}^j$ are bounds of x_i in the j^{th} direction.

4. PARTICLE SWARM OPTIMIZATION

PSO is an algorithm that is patterned after the social behavior of swarming such as fish schooling and bird flocking. It was proposed by Eberhart and Kennedy [29]. The initial process of PSO was improved by Eberhart and Shi [30].

PSO is a technique that improves an issue by repetitively attempting to come up with a more competitive solution, which is recognized as the best solution by the bee swarm. PSO resembles evolutionary computing; nonetheless, PSO is distinct because it does not include the evolution of operators.

Such evolution is replaced with the introduction of swiftness and position as well as the repetitive search of a certain location. Therefore, each competitive solution is presented in the form of a particle. Every particle has two characteristics: position x and velocity v, the x and v of the i^{th} particle may be supposed as follows:

$$x = x_{i1}, x_{i2}, \dots \dots , x_{in} \tag{7}$$

$$v = v_{i1}, v_{i2}, \dots \dots , v_{in} \tag{8}$$

where n represents the dimension of the problem,

$$v_i^j(t+1) = v_i^j(t) + c_1 \times r_1 \times \left[p_{i,j}^{best}(t) - x_i^j(t)\right] \\ +c_2 \times r_2 \times \left[g_{i,j}^{best}(t) - x_i^j(t)\right] \qquad (9)$$

where x represents the ith particle in the jth dimension at time step t, $v_i^j(t)$ is the swiftness of the ith particle in the jth dimension at time step t, $p_{i,j}^{best}(t)$ is the most appropriate individual solution of the ith particle in the jth dimension at time step t, $g_{i,j}^{best}(t)$ represents the most appropriate global location in the jth dimension at time step t, c_1 and c_2 are the positive constants called acceleration coefficients that are used to scale the addition of the cognitive and social components, and r_1 and r_2 are arbitrary numbers that are equally parted into the interval [0, 1]. In looking for a minimization problems, the distinctive most appropriate solution of the particle at time step $(t+1)$ is as follows:

$$p_{i,j}^{best}(t+1) = \begin{cases} p_i^{best}(t+1)\ if\ (x_i(t+1) \geq f(p_i^{best}(t))) \\ x_i^j(t)\ if\ (x_i(t+1) < f(p_i^{best}(t))) \end{cases} \qquad (10)$$

where $f(\,)$ is the impartial function, which is characteristic of a minimization problems; and $x_i(t+1)$ is the physical place of the ith particle at time step$(t+1)$. The latter can be calculated as follows:

$$x_i^j(t+1) = x_i^j(t) + v_i^j(t+1) \qquad (11)$$

Considering the minimization problems, the global best is identified using Equation (11) (N signifies the number of particles):

$$g^{best} = \min\{f(p^{best})\} \qquad (2)$$

In [30], Shi and Eberhart suggested adding a new indicator called inertia weight (w). This indicator is used to control the exploitation and exploration of the capabilities of a swarm. It controls the momentum of the particle by weighing the addition of the previous swiftness. When inertia weight ($\mathcal{W}$) is introduced, Equation (9) is transformed into the following form:

$$v_i^j(t+1) = w + v_i^j(t) + c_1 \times r_1 \times \left[p_{i,j}^{best}(t) - x_i^j(t)\right] \\ +c_2 \times r_2 \times \left[g_{i,j}^{best}(t) - x_i^j(t)\right] \qquad (13)$$

5. HYBRID ABC-PSO TRAINED FFNN FOR INTRUSION DETECTION SYSTEM

Recently, numerous researchers have employed heuristic algorithms in place of traditional algorithms to train FFNNS [19, 22]. Heuristic algorithms are found to be more efficient than traditional algorithms. When ANNs are used, the NN structure, which will be explained by the training algorithm, should be determined first. This article proposes using the new method presented in our previous article [31] to train an NN to classify intrusions in computer networks. The proposed algorithm is an ABC–PSO hybrid that is used, as previously stated, to discover suitable weights and biases in FFNNs and to avoid local minima. This hybrid algorithm is preferable because typical algorithms, such as BP, are frequently stuck in local minima and their results are not optimal. These drawbacks were also evident in our previous work, wherein a data set from the University of California, Irvine was tested for machine learning [32]. Consequently, we first used the proposed hybrid algorithm to perfect the connection between the weights of FFNNs. After this objective was realized, the perfected FFNN was utilized to categorize the KDD Cup 1999 data set.

A two-layer FFNN, in which n indicates the number of input nodes, h denotes the number of output nodes, and m signifies the number of hidden nodes in the system, may be observed. In this case, the KDD Cup 1999 is the best data set to utilize in assessing the proposed algorithm. The testing parameters are 41 input nodes and 5 output nodes given that we have assigned one to each class. The number of hidden nodes depends on the number of attributes and the class of each attribute, and thus, is determined by the following simple formula: (Attribute + Class)/2. In the hybrid method, a set of parameters are found as a result of some preliminary tests. With D as the dimension of population size, which is equal to 50, we can define the "limit" as ($SN \times D$). In a PSO method, we randomly generate r_1 and r_2 in Equation (9). The original velocities of the particles are larger than 0 and smaller than 1, c_1 and c_2 are equal to 2, and the number of iterations can be set to 500.

The hidden transfer function is assumed to be a sigmoid function, whereas the output transfer function is a linear activation function. Fitness in an FFNN can be determined using the fitness function that applies the error in the FFNN. An encoding method for the biases and weights of the FFNN are described in detail in our previous work [31].

6. DATA EVALUATION

The most popular and widely used data set for detecting intrusions and anomalies is the aforementioned KDD Cup 1999 data set [33, 34], which was created and developed in 1999 by Stolfo and other researchers [35]. It was built from information obtained from MIT Lincoln Laboratory, under the sponsorship of the Defense Advanced Research Projects Agency and the Air Force Research Laboratory. The data set comprises records totaling approximately 5 million. It represents TCP/IP packet connections, each of which contains 41 attributes (features), 38 of which are numeric and 3 are symbolic. The KDD Cup 1999 data set has 23 attacks, which have been categorized into four types of assault data [36]: denial of service (DOS), probing (PROBE), user to root (U2R), and remote to local (R2L). The set of attributes in KDD Cup 1999 is divided into three parts: basic, content, and traffic attributes. Basic attributes contain all attributes obtained from the package headers, whereas content attributes include all attributes extracted from the packet payload to detect a dubious behavior in the payload section. Traffic attributes are classified into two types, namely, "*Same Host*" and "*Same Service*," which respectively aid in determining whether an attribute is still connected with the same host or the same service [37].

The data set processing stage is the most important stage in our work. This stage is further divided into two substages. In the first substage, the data set that is used to evaluate algorithms is determined. We still use the KDD Cup 1999 data set for this task. However, the size of this data set is too large to be loaded into memory. Thus, we extracted a random sample of 149045 records from the data set. This random sample contains the following records: 87832 Normal, 55928 DOS, 4107 PROBE, 1126 R2L, and 52 U2R. Figure 2, shows the distribution of datasets and categories.

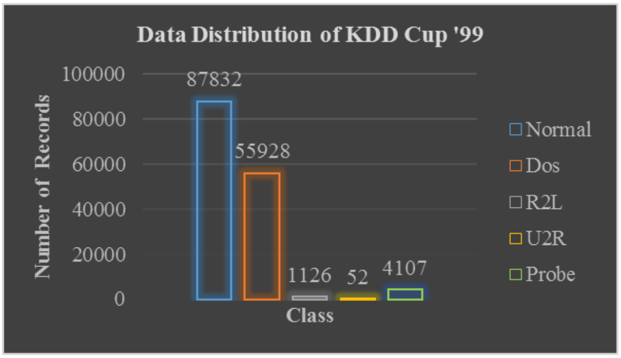

Figure 2: Data Distribution of KDD Cup '99

This random sample is divided into two subsamples. The first subsample is called the *training* data set, and the second subsample is called the *test* data set. The second sub stage of data set processing involves converting a symbolic value into a numeric value and then normalizing the numeric value. In this case, normalization indicates placing the numeric values within the same range. This phase is composed of the following steps. *Step 1*: Convert any symbolic value of KDD Cup 1999 into a numeric value. KDD Cup 1999 contains three attributes with symbolic values (Protocol Type, Service, and Flag). *Step 2*: Normalize the numeric value. The attributes of the data set have to be normalized before the data set is used in the training algorithm to provide regular semantics to the attribute values [38, 39]. To normalize numeric values into regular semantics between X_{min} and X_{max}, which are the minimum and maximum values of attribute *X*, respectively, X_{min} and X_{max} are first converted into a new range according to Equation (14) [40].

$$X_{new} = \frac{X_{current} - X_{min}}{X_{max} - X_{min}} \qquad (3)$$

From Equation (14), x represents the numeric value of the attribute, x_{min} represents the minimum value of the attribute to which x belongs to, x_{max} represents the maximum value of the attribute to which X belongs to, and $x_{current}$ represents the original range that is converted into a new value x_{new}.

7. EXPERIMENTAL RESULTS AND DISCUSSION

In this study, the experiment is performed in an Intel Core i5 2.3 GHz simulation platform with 4 GB RAM and 3 MB L2 cache. We have implemented the hybrid ABC–PSO optimized FFNN algorithm and compared its results with those of the other algorithms using JAVA and WEKA, which is an open source data mining software under JAVA environment. In the experiment, the heap size in JAVA is set to 3G.

The performance of IDS is measured and evaluated by many of measurement standards, but in this work we are committed to use the standards of measurement used by the WEKA tool. So we evaluate the proposed method against a category of artificial intelligence algorithms, implemented in WEKA tool. There are a few algorithms that are more important than others, and those are the Logistic one, the Radial basis, the Radial basis and the Multilayer perception, all of which are utilized in order to assess the performance of the suggested algorithm by testing this algorithms in comparison with the proposal by using the same parameters KDD Cup 99.

This study focuses on the performance of the four classification methods based on the previously mentioned performance measurement criteria. Figure 3 shows that compared with the other algorithms, the hybrid ABC–PSO optimized FFNN algorithm obtains the highest value, which is close to 1 in terms of kappa statistic. Furthermore, the following algorithms obtained different values in terms of kappa statistics in descending order: MLP, simple logistic regression, radial basis function NN, and voted perceptron.

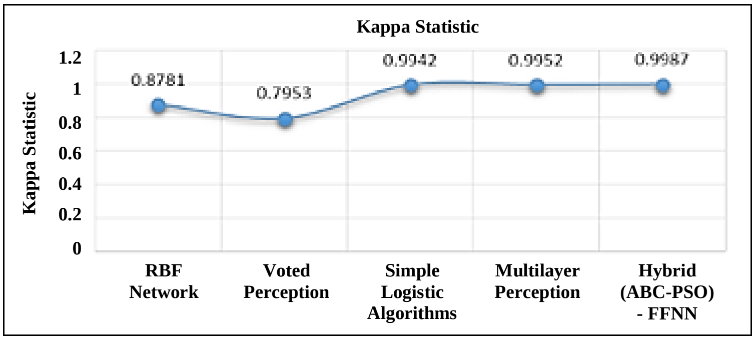

Figure 3: Kappa Statistic

As shown in Table 1, the hybrid ABC–PSO optimized FFNN algorithm, generated lower mean absolute error and root-mean-square error (RMSE) compared with the other algorithms, which generated high error values, as shown in Figure 4. Table 2 compares the performance of the five algorithms based on kappa statistic, mean absolute error, RMSE, relative absolute error, and root relative squared error.

Table 1
Comparison of Different Measurement Criteria

Algorithms	Kappa Statistic	Mean Absolute Error	Root Mean Squared Error	Relative Absolute Error	Root Relative Squared Error
RBF Network	0.8781	0.0390	0.1388	19.0928%	43.4252%
Voted Perceptron	0.7953	0.0451	0.2081	22.063%	65.1012%
Simple Logistic	0.9942	0.003	0.0325	1.4466%	10.1655%
Multilayer Perceptron	0.9952	0.017	0.0298	0.8111%	9.3083%
Hybrid ABC-PSO Optimized FFNN	0.9987	0.0017	0.0159	0.3558%	4.9758%

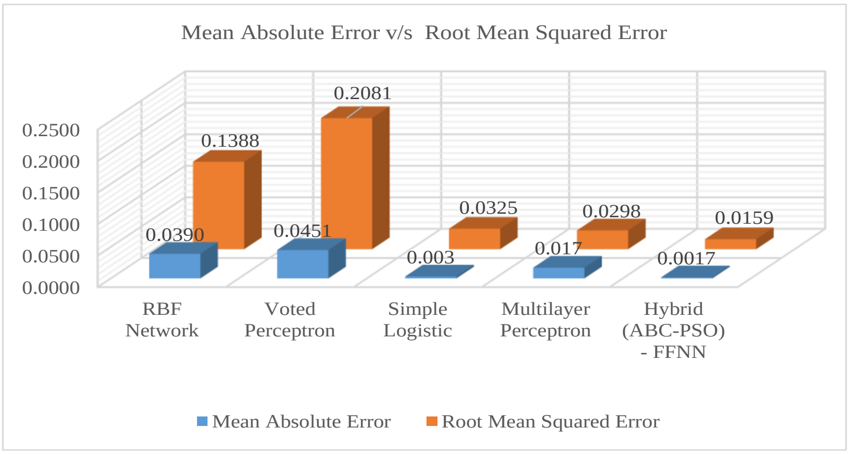

Figure 4: Mean Absolute Error v/s Root Mean Squared Error

Furthermore, the hybrid ABC–PSO optimized FFNN algorithm obtained lower relative absolute error and root relative squared error compared with the other algorithms, which achieved high error values, as shown in Figure 5.

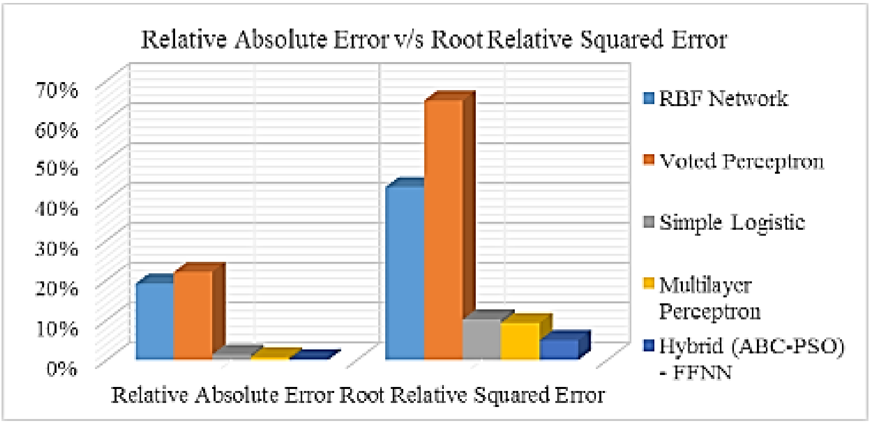

Figure 5: Relative Absolute Error v/s Root Relative Squared Error

Table 2 shows the performance accuracies of the five compared algorithms. The experiment results indicate that the hybrid ABC–PSO optimized FFNN algorithm overcomes all other algorithms, as shown in Figure 6.

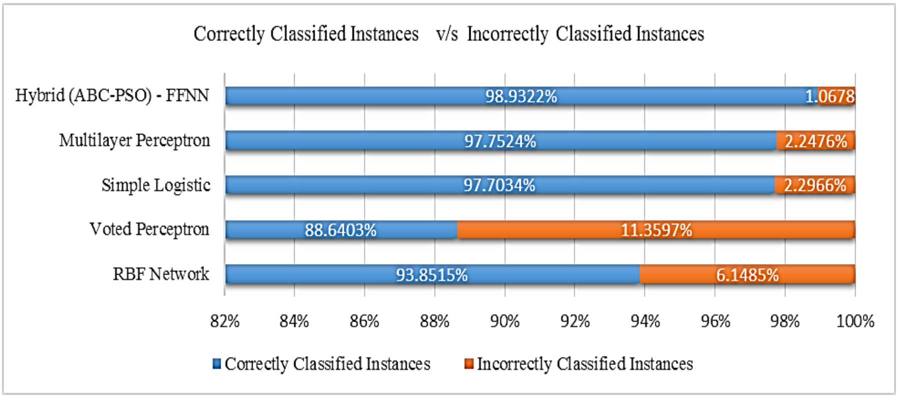

Figure 6: Correctly Classified Instances v/s Incorrectly Classified Instance

Table 2
Correctly Classified Instances v/s Incorrectly Classified Instances

Algorithms	Correctly Classified	Incorrectly Classified	Correctly Classified Instances	Incorrectly Classified Instances
RBF Network	93.8515%	6.1485%	139881	9164
Voted Perceptron	88.6403%	11.3597%	132114	16931
Simple Logistic	97.7034%	2.2966%	145622	3423
Multilayer Perceptron	97.7524%	2.2476%	145695	3350
Hybrid (ABC-PSO) Optimized FFNN	98.9322%	1.0678%	147453	1592

8. CONCLUSION AND FUTURE WORK

In this work, we introduced a new approach for intrusion detection in a network using a hybrid algorithm, namely, ABC–PSO, which was used to optimize the interconnection weights of the FFNN algorithm. We used an FFNN with one hidden layer for network anomaly detection; the hidden layer was, in turn, used to classify and detect anomalous packets. The results showed that the proposed method exhibited high accuracy in classifying normal and abnormal records (packages) in a data set (KDD Cup 1999). The results of the proposed method were compared with those of four other NN algorithms incorporated in the WEKA toolkit. The results showed the superiority of the proposed method over the other algorithms. In a future study, we will expand the application of the hybrid algorithm by using it to select some attributes of the data set to improve detection performance.

ACKNOWLEDGMENT

This research experiment has been funded by Universiti Sains Malaysia under the Research University Cluster Grant Scheme (RUC) for “Reconstruction of the Neural

Micro circuitry or Reward-Controlled Learning in the Rat Hippocampus” [1001/PKOMP/8630022]. The research also partially supported by the Fundamental Research Grant Scheme (FRGS) for "Content-Based Analysis Framework for Better Email Forensic and Cyber Investigation” [203/PKOMP/6711426], also support from USM Fellowship [APEX (308/AIPS/415401)] from Institute of Postgraduate Studies, Universiti Sains Malaysia.

REFERENCES

1. Anderson, J.P. (1980). *Computer security threat monitoring and surveillance* (Vol. 17). Technical Report, James P. Anderson Company, Fort Washington, Pennsylvania.
2. Denning, D.E. (1987). An intrusion-detection model. *IEEE Transactions on Software Engineering*, 2, 222-232.
3. Ghanem, W.A.H. and Belaton, B. (2013). Improving accuracy of applications fingerprinting on local networks using NMAP-AMAP-ETTERCAP as a hybrid framework. In *IEEE International Conference on Control System, Computing and Engineering (ICCSCE)*, 403-407.
4. Lazarevic, A., Kumar, V. and Srivastava, J. (2005). Intrusion detection: A survey. In *Managing Cyber Threats* (pp. 19-78). Springer US.
5. Lu, H., Setiono, R. and Liu, H. (1996). Effective data mining using neural networks. *IEEE Transactions on Knowledge and Data Engineering*, 8(6), 957-961.
6. Bishop, C.M. (2006). *Pattern recognition and machine learning*. Springer.
7. Ozturk, C. and Karaboga, D. (2011). Hybrid artificial bee colony algorithm for neural network training. In *2011 IEEE Congress on Evolutionary Computation (CEC),* 84-88.
8. Yao, X. (1999). Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9), 1423-1447.
9. Garro, B., Sossa, H. and Vázquez, R. (2011). Artificial neural network synthesis by means of artificial bee colony (abc) algorithm. In *2011 IEEE Congress on Evolutionary Computation (CEC)*, 331-338.
10. Horne, B.G. (1993). Progress in supervised neural networks. *Signal Processing Magazine*, IEEE, 10(1), 8-39.
11. Karaboga, D., Akay, B. and Ozturk, C. (2007). Artificial bee colony (ABC) optimization algorithm for training feed-forward neural networks. In *Modeling Decisions for Artificial Intelligence* (pp. 318-329). Springer Berlin Heidelberg.
12. Carvalho, M. and Ludermir, T.B. (2006). Hybrid training of feed-forward neural networks with particle swarm optimization. In *Neural Information Processing* (pp. 1061-1070). Springer Berlin Heidelberg.
13. Meissner, M., Schmuker, M. and Schneider, G. (2006). Optimized Particle Swarm Optimization (OPSO) and its application to artificial neural network training. *BMC bioinformatics*, 7(1), 125.
14. Yang, X.S. (2010). A new metaheuristic bat-inspired algorithm. *Nature inspired cooperative strategies for optimization* (NICSO 2010), 65-74.
15. Ghanem, W.A. and Jantan, A. (2017). An enhanced Bat algorithm with mutation operator for numerical optimization problems. *Neural Computing and Applications,* 1-35.

16. Ryan, J., Lin, M.J. and Miikkulainen, R. (1998). Intrusion detection with neural networks. *Advances in Neural Information Processing Systems*, 943-949.
17. Hornik, K., Stinchcombe, M. and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5), 359-366.
18. Malakooti, B. and Zhou, Y. (1998). Approximating polynomial functions by feedforward artificial neural networks: capacity analysis and design. *Applied Mathematics and Computation*, 90(1), 27-51.
19. Kıran, M.S. and Gündüz, M. (2013). A recombination-based hybridization of particle swarm optimization and artificial bee colony algorithm for continuous optimization problems. *Applied Soft Computing*, 13(4), 2188-2203.
20. Lin, C.J., Chen, C.H. and Lee, C.Y. (2004). A self-adaptive quantum radial basis function network for classification applications. In *Proceedings of 2004 IEEE International Joint Conference on Neural Networks*, 4, 3263-3268.
21. Isa, N.A.M. and Mamat, W.M.F.W. (2011). Clustered-Hybrid Multilayer Perceptron network for pattern recognition application. *Applied Soft Computing*, 11(1), 1457-1466.
22. Zhang, J.R., Zhang, J., Lok, T.M. and Lyu, M.R. (2007). A hybrid particle swarm optimization–back-propagation algorithm for feedforward neural network training. *Applied Mathematics and Computation*, 185(2), 1026-1037.
23. Karaboga, D. (2005). *An idea based on honey bee swarm for numerical optimization* (Vol. 200). Technical report-tr06, Erciyes University, Engineering Faculty, Computer Engineering Department.
24. Karaboga, D. and Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. *Journal of Global Optimization*, 39(3), 459-471.
25. Karaboga, D. and Ozturk, C. (2009). Neural networks training by artificial bee colony algorithm on pattern classification. *Neural Network World*, 19(3), 279.
26. Ghanem, W.A. and Jantan, A. (2016). Hybridizing artificial bee colony with monarch butterfly optimization for numerical optimization problems. *Neural Computing and Applications*, 1-19.
27. Bansal, J.C., Sharma, H. and Jadon, S.S. (2013). Artificial bee colony algorithm: A survey. *International Journal of Advanced Intelligence Paradigms*, 5(1-2), 123-159.
28. Bolaji, A.L.A., Khader, A.T., Al-Betar, M.A. and Awadallah, M.A. (2013). Artificial bee colony algorithm, its variants and applications: A survey. *Journal of Theoretical & Applied Information Technology*, 47(2), 434-459.
29. Kennedy, J. (2010). Particle swarm optimization. In *Encyclopedia of Machine Learning* (pp. 760-766). Springer US.
30. Shi, Y. and Eberhart, R. (1998). A modified particle swarm optimizer. In *IEEE International Conference on Evolutionary Computation Proceedings, 1998 IEEE World Congress on Computational Intelligence*, 69-73.
31. Ghanem, W.A.H. and Jantan, A. (2014). Using hybrid artificial bee colony algorithm and particle swarm optimization for training feed-forward neural networks. *Journal of Theoretical and Applied Information Technology*, 67(3), 664-674.
32. Ghanem, W.A.H. and Jantan, A. (2014). Swarm intelligence and neural network for data classification. In *2014 IEEE International Conference on Control System, Computing and Engineering* (ICCSCE), (pp. 196-201).

33. Tavallaee, M., Bagheri, E., Lu, W. and Ghorbani, A. A. (2009). A detailed analysis of the KDD CUP 99 data set. In *Proceedings of the Second IEEE Symposium on Computational Intelligence for Security and Defence Applications* 2009.
34. KDD,KDDCup, http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html. Accessed Apr 2014.
35. Stolfo, S.J., Fan, W., Lee, W., Prodromidis, A. and Chan, P.K. (2000). Cost-based modeling for fraud and intrusion detection: Results from the JAM project. In *IEEE DARPA Information Survivability Conference and Exposition*, 2000. DISCEX'00. Proceedings (Vol. 2, pp. 130-144).
36. Siddiqui, M.K. and Naahid, S. (2013). Analysis of KDD CUP 99 dataset using Clustering based Data Mining. *International Journal of Database Theory and Application*, 6(5), 23-34.
37. Ganapathy, S., Kulothungan, K., Muthurajkumar, S., Vijayalakshmi, M., Yogesh, P. and Kannan, A. (2013). Intelligent feature selection and classification techniques for intrusion detection in networks: a survey. *EURASIP Journal on Wireless Communications and Networking*, 2013(1), 1-16.
38. Sindhu, S.S.S., Geetha, S. and Kannan, A. (2012). Decision tree based light weight intrusion detection using a wrapper approach. *Expert Systems with Applications*, 39(1), 129-141.
39. Wang, W., Zhang, X., Gombault, S. and Knapskog, S.J. (2009). Attribute normalization in network intrusion detection. In *IEEE 10th International Symposium on Pervasive Systems, Algorithms, and Networks* (ISPAN), 2009 (pp. 448-453).
40. Ibrahim, L.M., Basheer, D.T. and Mahmod, M.S. (2013). A comparison study for intrusion database (Kdd99, Nsl-Kdd) based on self-organization map (SOM) artificial neural network. *Journal of Engineering Science and Technology*, 8(1), 107-119.