Soft Comput
DOI 10.1007/s00500-015-1967-z

METHODOLOGIES AND APPLICATION

A study on intrusion detection using neural networks trained with evolutionary algorithms

Tirtharaj Dash[1,2]

Abstract Intrusion detection has been playing a crucial role for making a computer network secure for any transaction. An intrusion detection system (IDS) detects various types of malicious network traffic and computer usage, which sometimes may not be detected by a conventional firewall. Recently, many IDS have been developed based on machine learning techniques. Specifically, advanced detection approaches created by combining or integrating evolutionary algorithms and neural networks have shown better detection performance than general machine learning approaches. The present study reports two new hybrid intrusion detection methods; one is based on gravitational search (GS), and other one is a combination of GS and particle swarm optimization (GSPSO). These two techniques have been successfully implemented to train artificial neural network (ANN) and the resulting models: GS-ANN and GSPSO-ANN are successfully applied for intrusion detection process. The applicability of these proposed approaches is also compared with other conventional methods such as decision tree, ANN based on gradient descent (GD-ANN), ANN based on genetic algorithm (GA-ANN) and ANN based on PSO (PSO-ANN) by testing with NSL-KDD dataset. Moreover, the results obtained by GS-ANN and GSPSO-ANN are found to be statistically significant based on the popular Wilcoxon's rank sum test as compared to other conventional techniques. The obtained test results reported that the proposed GS-ANN and GSPSO-ANN could achieve a maximum detection accuracy of 94.9 and 98.13 % respectively. The proposed models (GS-ANN and GSPSO-ANN) could also achieve good performance when tested with highly imbalanced datasets.

Keywords Intrusion detection · Intrusion detection system · Artificial neural network · NSL-KDD

Communicated by V. Loia.

Electronic supplementary material The online version of this article (doi:10.1007/s00500-015-1967-z) contains supplementary material, which is available to authorized users.

✉ Tirtharaj Dash
tirtharaj@goa.bits-pilani.ac.in

[1] Data Science Laboratory, School of Computer Science, National Institute of Science and Technology, Berhampur 761008, India

[2] Present Address: Department of Computer Science and Information Systems, Birla Institute of Technology and Science Pilani, K.K. Birla Goa Campus, Goa 403726, India

1 Introduction

Internet technology has gained its applications in many different areas in our life such as online transactions, online auctions, internet banking, and online examinations. However, due to the weakness of server or the strength of the hacker, network has often been intruded with denial-of-service (DoS) or distributed DoS (DDoS) attacks (Yu et al. 2007; Horng et al. 2011). Password protection, firewalls, access control and encryption have been used as traditional intrusion prevention techniques for last several years. However, these techniques have failed to fully protect networks and systems from increasingly sophisticated attacks and malwares (Wu and Banzhaf 2008). Recently, intrusion detection system (IDS) has become an essential component of cyber security infrastructures and it is applied to protect network and systems from the above-mentioned threats. IDS aims to solve the problem of intrusion detection by detecting malicious attacks such as DoS, port scans and even try to access computer resources by monitoring network traffic (Wu and

Published online: 07 December 2015

Banzhaf 2008). ID can be achieved both by signature recognition and anomaly detection (Ye et al. 2002). It is learned that signature recognition models store the patterns of intrusion signatures and compare these patterns with observed activities to find a possible match. If an optimal match is found, then an intrusion is being detected. However, the signature recognition models are incapable to detect novel attacks due to their unknown patterns. In the other hand, anomaly detection (AD) models identify an intrusion only when there is a large deviation from the normal profile built on long-term normal activities (Sindhu et al. 2009).

It has also been reported that there can be three different types of IDS that are based on network architecture: (i) host-based, (ii) network-based and (iii) distributed. Host-based IDS are implemented on the host computer system which uses activity logs and network traffic records of the single host for data analysis and processing for the ID. The disadvantage of such ID is that it is incapable of detecting simultaneous and multiple attacks because their data can be found in the host logs or records. Network-based IDS is a dedicated computer system or a special hardware architecture in which ID software is installed. Such ID can detect attacks against multiple hosts on a single subnet, but it usually cannot monitor multiple subnets at one time. Distributed detection system usually allows the software module to be installed in all the computers and one of these networked computer acts as a central controller which collects and analyses data obtained from other computers. However, major demerit of this type of IDS is that the central computer often remains vulnerable to intrusion.

When building IDS, one needs to take care of many issues, such as data collection, data preprocessing, intrusion recognition, reporting the instance, and providing alarm or response. Among these mentioned issues, intrusion recognition is at the core of IDS, where incoming data instance is examined and compared with patterns of intrusive behavior so that both successful and unsuccessful intrusion attempts can be identified (Patcha and Park 2007). It is well known that, if the identification accuracy is at high level then the network is more secure. Hence, many researchers in the last decade have reported to enhance the ID accuracy (Ye et al. 2002; Sindhu et al. 2009; Patcha and Park 2007; Wu and Yen 2009; Chen et al. 2007; Toosi and Kahani 2007; Vollmer et al. 2011). In recent studies, techniques such as artificial neural network (ANN), decision tree (DT), evolutionary algorithms [(e.g., genetic algorithm (GA), particle swarm optimization (PSO), simulated annealing (SA)], rule-based data mining, swarm intelligence have gained significance attention to solve the problem of intrusion detection.

However, the main shortcomings of ANN-based IDS exist in two aspects: (i) lower detection precision for lower frequent attacks viz. User to Root (U2R), Remote to Local (R2L) and (ii) stability of IDS for larger patterns (Beghdad 2008). The main reason of these weaknesses is that the size of attack patterns in the dataset is imbalanced and ANN is seldom capable of regenerating target result when trained with small number of input patterns. Therefore, it is not possible for the model to give precise results for low frequent attacks. Furthermore, these low frequent attacks are considered to be outliers in IDS and hence the ANN gets trapped in the local minima of the evolution curve when trained with gradient descent (GD) search. Furthermore, if the training dataset is more biased towards a particular class (meaning, majority of instances in the dataset belongs to a certain class), then the ANN is prone to over-fitting. This can be a common scenario in intrusion detection if more frequent attacks are present in the ANN training dataset, whereas same attacks are not found during the testing or deployment phase of the IDS, there is high possibility that the detection system will give unexpected outcomes viz. false alarming or blocking of traffic.

Motivation behind this work Three important factors are responsible towards working on this area of research. *First,* the need of a IDS; *second,* applicability of IDS in real life, and *third*, present deficiencies in reported IDS. Now-a-days, with increase in social networking, computer systems most commonly store sensitive information and some present software systems often contain vulnerabilities which could be one of the weaknesses of the host computer system and strength of the possible attackers. Coming to real life, not all systems are computer based. There are few small self-sustained electronic systems which are also pretty useful in daily real life viz. car alarms, house alarms, fire detectors, flood warning systems or earthquake detection systems. These systems are also sometimes prone to intrusions which may not always be due to attackers but may due to sensed data at different times. In the former discussions in the literature, some problems are still there in current IDS research outcomes which need to be addressed effectively. However, it should be noted that resolving all the existing problems in this area has not been an easy task and hence, it may also be worth mentioning that a small contribution in this field could probably play an important role in the field of applicability of such machine learning-based methods.

In this work, an attempt has been made to solve some of the aforementioned problems using a combination of ANN and some of the newer evolutionary algorithms. This work combines ANN with using a new and most widely used evolutionary algorithm called gravitational search (GS) (Rashedi et al. 2009) and henceforth the developed ANN can be recognized as GS-ANN. However, in some of the research works, it has also been reported that the GS algorithm is slower as compared to other swarm-based algorithms such as PSO (Eberhart and Kennedym 1995). And, a hybrid algorithm called GSPSO has been used for training the ANN. This model can be recognized as GSPSO-ANN. It should be noted

that the combination of GS and PSO algorithms has been used to train a neural network for different real-world datasets in Mirjalili et al. (2012) and Dash et al. (2015a) and have been proved to be successful in its applications. To illustrate the applicability of the proposed approaches, the performance measures, e.g., error, training time, decision time, average detection accuracy rate have been obtained by testing with NSL-KDD dataset (Tavallaee et al. 2009), and the results have been compared with popularly used techniques such as DT, GD-ANN, GA-ANN, and PSO-ANN.

The rest of the paper is organized as follows. The Sect. 2 gives a brief description of related works in the field of ID using computational intelligence (CI). In Sect. 3, GS and GSPSO-based algorithm for IDS has been demonstrated. For evaluation of the proposed approach, Sect. 4 gives a detailed description of data collection, preprocessing, results and discussion, etc. Finally, Sect. 5 draws the conclusions and future research directions.

2 A brief review of recent related works

In recent years, several reports have described that the ID can be both misuse detection and anomaly detection. Patcha and Park (Patcha and Park (2007)) and Lazarevic et al. (Lazarevic et al. (2005)) categorized these various approaches into following types.

Misuse-based methods These methods are reported to be efficient for detecting known attacks, but not capable of unknown attacks. This is again sub-categorized into following types: (i) pattern recognition and machine learning methods; which uses signature patterns for basis of detection of known or unknown threats (Mahoney and Chan 2002; Kumar and Selvakumar 2011), (ii) implication rules-based methods: uses a rule base for implementing the incident detector module, (iii) data mining techniques: uses ensemble of neural networks, swarm intelligence, hybrid of soft computing approaches, Principal Component Analysis (PCA) and many other techniques (Kumar and Selvakumar 2011).

Anomaly-based methods This method analyses the profiles that represent normal traffic behavior. Here, an anomaly detector is used to create a base profile of the normal traffic activity. Any traffic pattern which deviates from the normal pattern is treated as an anomaly. There are two major techniques used in this method: (i) statistical method (uses two profiles; current and previous based on statistical analysis. Incoming connection is checked against previously stored profiles, if mismatch is found, the current profile is updated (Gu et al. 2005)], and (ii) machine learning and data mining techniques (Catania and Garino 2012).

The data mining techniques have been reported to be superior to other available non-data mining-based techniques both for misuse detection and anomaly detection in intrusion detection. Some of such works in the available literature have been reported here. Horng et al. proposed a hybrid of hierarchical clustering and Support Vector Machine (SVM) for intrusion detection. They used the hierarchical clustering for extracting important features from the KDD Cup'1999 dataset (KDD 1999) and then SVM was applied for detection of possible intrusions. However, their model was not capable of detecting attacks with new patterns (Horng et al. 2011). Fuzzy classifier and its hybrid systems have been successfully employed for efficient detection of intrusion (Marín-Blázquez and Pérez 2009). Wang et al. tried to improve the detection accuracy for unknown attacks by hybridizing fuzzy clustering with ANN (FC-ANN). The fuzzy clustering module was used to extract subset of patterns from the whole dataset and a number of ANNs were used as the base models for checking unknown patterns. Finally, the results of all the ANN modules were aggregated for finding the final results. However, their work was not able to detect probe attacks with high precision (Wang et al. 2010). Similar work has also been carried out by Aydin et al. who combined packet header anomaly detection (PHAD), network traffic anomaly detection (NETAD) and SNORT. In their work, the former two anomaly detection approaches were used as preprocessor and SNORT was used for signature pattern matching (Aydin et al. 2009). Evolutionary multi-objective optimization techniques have also been used for designing efficient IDS. In Ramasubramanian and Kannan (2006), researchers used neuro-genetic approach for designing IDS. Similar application of multi-objective optimization can be seen in Gómez et al. (2013), where the optimization technique was used for automatic rule generation in signature-based IDS.

In last few years, techniques such as hidden Markov model, nonlinear analysis, Boltzmann machine and few other statistical techniques have also gained popularity with regard to their application in network intrusion detection (Manikopoulos and Papavassiliou 2002; Palmieri and Fiore 2010; Zbilut and Webber 2006; Rastegari et al. 2015; Dainotti et al. 2008; Fiore et al. 2013). Manikopoulos and Papavassiliou proposed a general-purpose hierarchical multitier multiwindow statistical anomaly detection technology which can work in wired as well as wireless ad hoc networks. The authors used a combination of statistical model and multivariate classifier as their tool which could detect attacks with traffic anomaly intensity as low as 3–5 % (Manikopoulos and Papavassiliou 2002). In another study, researchers have used nonlinear analysis towards detection of network anomaly. Their work used recurrence quantification analysis (a popular technique in the domain of science to explore statistical time series (Zbilut and Webber 2006)] to study and interpret recurrence patterns which originated by complex traffic (Palmieri and Fiore 2010). In a very recent work, Rastegari et al. tried to use GA and interval-based rules for

intrusion detection. The evaluation of their new approach with three datasets viz. NSL-KDD, and another dataset constructed based on MIT Lincoln Laboratory normal traffic and the low-rate DDoS attack scenario from CAIDA demonstrated better detection performance (Rastegari et al. 2015). Researchers in a study introduced a combination of PCA and chaotic PSO for improving performance of multilayer SVM for intrusion detection (Kuang et al. 2015). This model demonstrated higher accuracy with comparatively smaller computational time. In few other works, researchers have tried to identify distributed intrusions in the network by incorporating models such as independent component analysis (Palmieri et al. 2014), data mining techniques integrated with common intrusion detection framework (Lee et al. 2000).

Network infrastructure is also found as a crucial issue while designing IDS. Various IDS models for mobility networks such as wireless sensor network (WSN), mobile ad hoc network (MANET), vehicular ad hoc network (VANET) based on neural network and fuzzy logic approaches have also been reported in many of the research findings and can be obtained in the references (Linda et al. 2009; Hu et al. 2009; Mabu et al. 2011; Shin et al. 2010; Bao et al. 2012; Shakshuki et al. 2013). Linda et al. proposed a IDS for critical infrastructure networks where the model was build based on neural network. They used window-based feature extraction strategy and randomly generated the intrusion detection vectors. Their model was trained with two learning algorithms, error-back propagation and Levenberg–Marquardt algorithm (Linda et al. 2009). Hidden Markov model has also been used for anomaly-based ID which used system calls during detection (Hu et al. 2009). IDS have been designed for industrial WSN using different level of clustering in node architectures and hops (Shin et al. 2010; Bao et al. 2012).

3 Methodology

This section depicts the GS-ANN and GSPSO-ANN approaches for designing the IDS. The first subsection presents the whole framework of the designed IDS and the proposed techniques have been discussed in subsequent subsections.

3.1 Framework of the designed IDS

Figure 1 shows the framework of the IDS designing steps and it can be seen that these steps can be grouped into four major phases/modules which have been described as follows. However, the work of each module in the given framework has been described in the "Experimental results" section (i.e., Sect. 4) for further understanding.

Data input phase The dataset (training and testing) will be given as input to the next phase for various processing by the ANN module.

ANN design phase From the input data, number of training attributes (N) will be obtained which is the number of inputs to the ANN. It should be noted that ANN designed in this work is a multilayered ANN (classically known as multilayered perceptron (MLP) which is usually a multilayered (one input layer–one hidden layer–one output layer) feedforward neural network) of architecture $N{:}M{:}1$ where M is the number of nodes in the hidden layer (Dash et al. 2015b). The inter-layer weights of the ANN are the input layer to hidden layer weights (which is represented as a set V), and the hidden layer to output layer weights (which is represented as a set W). Hence, as per designed architecture, there are $N \times M$ number of weights. Similarly, there is a single bias in the output layer and M biases in the hidden layer. It is worth mentioning that the number of output (also called prediction) node is kept to one. And, during the prediction or testing phase, a threshold was selected based on which a wise decision could be taken on the processed input instance.

Training of ANN with dataset The above-mentioned ANN will be trained with considered training dataset to update the knowledge base (weights viz. V, W and biases) with GS or GSPSO algorithm. These algorithms have been used for two phases of ANN learning. First, these are used for setting initial synaptic weights (V, W and biases) in each layer; second, these are used for updating these synaptic weights after each epoch. It should be noted that as there are two stages of processing which are (i) input layer to hidden layer processing, and (ii) hidden layer to output layer processing. In both hidden and output layer, the sigmoid activation function (Eq. 1) is used for transforming net input to a node to output from that node.

$$f_{\text{sig}}(x) = \frac{1}{1 + \mathrm{e}^{-x}} \tag{1}$$

In this work, mean squared error (MSE) is chosen as the objective function for minimization by the evolutionary algorithms and can be represented as Eq. (2), where NoP is the number of patterns/instances in the training dataset, output_i is computed output for the current pattern instance i; and target_i is the corresponding target for the instance i. It should be noted that there are few other training evaluation measures such as room mean squared error (RMSE), mean absolute percentage error (MAPE), mean absolute error (MAE), and mean percentage error (MPE) which researchers sometimes use in many of their works. However, this work uses of MSE as principal objective function which is computed epoch wise.

$$\text{MSE} = \frac{1}{\text{NoP}} \sum_{i=1}^{\text{NoP}} (\text{output}_i - \text{target}_i)^2 \tag{2}$$

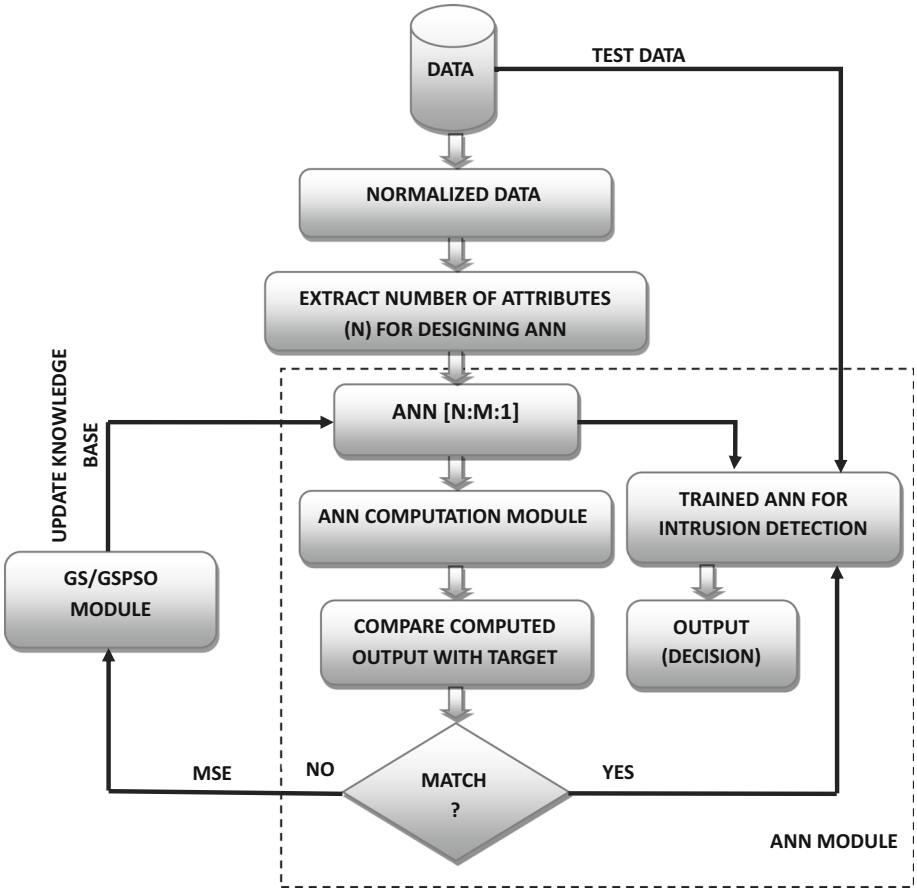

Fig. 1 Framework of the proposed IDS

Testing of ANN After the ANN is trained with the training dataset, each instance of the test dataset will be given as input to the ANN for prediction. In this testing phase, the predicted output will be checked with the closest match with any of the target class. Based on this output class, selective action can be taken for the currently tested instance.

3.2 Training of ANN with GS and GSPSO algorithms

It should be noted that the entirely presented evolutionary algorithm-based approaches such as GA-ANN, PSO-ANN, GS-ANN and GSPSO-ANN optimizes the neural network training based on the initial weight and biases of the ANN. Hence, the inputs to these algorithms will be weight set (V, W and biases) of the ANN. However, as DT, GD-ANN, GA-ANN and PSO-ANN have been implemented numerous times in the literatures, this paper does not give more details on these approaches. The procedure of training ANN with GS and its combination with PSO can be seen in Mirjalili et al. (2012) and Dash et al. (2015a). However, for better readability and clarity of the readers, this subsection describes the GSPSO-ANN algorithm. It should be noted that the hybrid algorithm, GSPSO which is used to train the ANN for the designed IDS is based on a combined concept of local search capability of GS (Rashedi et al. 2009) with social movement behavior of PSO (Eberhart and Kennedym 1995; Mirjalili et al. 2012). The steps in GSPSO algorithm are a combination of steps involved in GS algorithm and the additional steps are the steps involved in updating velocity and position in PSO algorithm. These steps are presented as an algorithm in Algorithm (1).

Algorithm 1 Steps involved in GSPSO algorithm:

```
procedure GSPSO–ALGORITHM
    Initialize all agents (position, mass, acceleration)
    while Stopping criteria not satisfied do
        Calculate gravitational force acting on each agent due to mutual attraction with neighbor agents
        Calculate acceleration of each particle due to force of attraction
        Update the best solution with new position of the agents (candidate solutions)
    end while
end procedure
```

Mathematically, the GSPSO system could be considered as an isolated system of agents similar to GS algorithm which is like a small artificial world of masses obeying the Newtonian laws of gravitation and motion. More precisely,

masses or agents obey the two fundamental scientific laws: Law of gravity and Law of motion (Rashedi et al. 2009).

Now, consider a system with N agents. We define the position of the ith agent by:

$$X_i = (x_i^1, x_i^2, \ldots, x_i^d, \ldots, x_i^n) \quad \text{for} \quad 1 \leq i \leq N$$

where x_i^d represents the position of ith agent in the dth dimension.

At a time t, the gravitational force acting on mass i from mass j can be defined as follows.

$$F_{ij}^d = G(t)\frac{M_{pi}(t) \times M_{aj}(t)}{R_{ij}(t) + \epsilon}(x_j^d(t) - x_i^d(t)) \tag{3}$$

In the Eq. (3), $G(t)$ is the gravitational constant at a time step t, ϵ is a small constant, M_{pi} is the passive gravitational mass related to agent i, M_{aj} is the active gravitational mass related to agent j. The value of G and masses of agents as function of time can be obtained using the following equations.

The gravitational constant, G can be represented as, $G(t)$ where G_0 is the initial gravitational constant and T is the maximum iteration.

$$G(t) = G_0 \times \mathrm{e}^{\frac{-\eta \times t}{T}} \tag{4}$$

While solving a global minimization problem, fitness value fit_i is computed for all feasible agent positions in X where $i = 1, 2, \ldots, N$ and N is number of agents in the search space. The fitness value is used to find masses of the agents using following equations, where best(t) and worst(t) represents the minimum and maximum fitness values. In this work, it is assumed that gravitational mass and inertia mass are equal and hence the following equality condition is considered and further computations have been made.

$$M_{ai} = M_{pi} = M_{ii} = M_i; i = 1, 2, \ldots, N \tag{5}$$

where M_{ii} represents the inertial mass of an agent i and M_i represents the gravitational mass of an agent i.

$$m_i(t) = \frac{\mathrm{fit}_i(t) - \mathrm{worst}(t)}{\mathrm{best}(t) - \mathrm{worst}(t)} \tag{6}$$

$$M_i(t) = \frac{m_i(t)}{\sum_{j=1}^{N} m_j(t)} \tag{7}$$

In the above equations, $\mathrm{fit}_i(t)$ represents the fitness value of the agent i at time t. For a global minimization problem, the best(t) and the worst(t) can be mathematically represented as,

$$\mathrm{best}(t) = \min\{\mathrm{fit}_j(t)\}; \quad \forall j \epsilon \{1, 2, \ldots, N\} \tag{8}$$

$$\mathrm{worst}(t) = \max\{\mathrm{fit}_j(t)\}; \quad \forall j \epsilon \{1, 2, \ldots, N\} \tag{9}$$

R_{ij} is the Euclidean distance between two entities i and j and is defined as given following equations.

$$R_{ij}(t) = \| X_i(t), X_j(t) \|_2 \tag{10}$$

$$R_{ij}(t) = \left(\sum_{p=1}^{n}\left(X_i^p(t) - X_j^p(t)\right)^2\right)^{\frac{1}{2}} \tag{11}$$

To give a stochastic characteristic to the GSPSO algorithm, it is considered that the total force that acts on agent i in a dimension d be a randomly weighted sum of dth components of the forces exerted from other agents:

$$F_i^d(t) = \sum_{j=1, j\neq i}^{N} \alpha_j F_{ij}^d(t) \tag{12}$$

where α_i is a small random number in the range [0, 1] to give a randomized characteristic to the search. The new acceleration of each agent i at a time t can be obtained using this field F_i and mass of an agent.

$$a_i^d(t) = \frac{F_i^d(t)}{M_i(t)} \tag{13}$$

Velocity and position of an agent i are calculated using the following equations, respectively; where w is a weighting function, $V_i(t)$ is the velocity of agent i at time t, c_1' and c_2' are the acceleration coefficients, $gBest$ is the best fitness so far, $X_i(t)$ is the position of agent i at time t; a_i is the acceleration of agent i, θ_1 and θ_2 are random numbers in the range [0,1].

$$\chi_1 = \left(c_1' \times \theta_1 \times a_i \times c_i(t)\right) \tag{14}$$

$$\chi_2 = \left(c_2' \times \theta_2 \times (gBest - X_i(t))\right) \tag{15}$$

$$V_i(t) = (w \times V_i(t)) + \chi_1 + \chi_2 \tag{16}$$

The above described GSPSO algorithm is used to train ANN by considering its weights (V, W and biases) as input parameters which are to be tuned. The training algorithm based on GSPSO algorithm has been given in the form of an algorithm (see Algorithm (2)). The algorithmic or theoretical time complexity of the presented algorithm has been given in Appendix section.

4 Experimental results

To evaluate the performances of GS-ANN and GSPSO-ANN approach, a series of experiments on NSL-KDD dataset were conducted. These experiments are also conducted using DT, GD-ANN, PSO-ANN for comparative studies. Results obtained from each models are compared with respect to their intrusion detection accuracy and simulation time as

illustrated in upcoming subsections. These proposed methods were implemented and evaluated in MATLAB R2010a on Windows PC with Quad-Core 3.30 GHz CPU and 4 GB RAM.

Algorithm 2 GSPSO–ANN for Intrusion Detection

```
procedure GSPSO-ANN-ID
    Input the Intrusion dataset (training and testing)
    Normalize the input dataset
    Initialize ANN parameters based on input data matrix: N, M, weights (V, W and biases)
    Initialize training parameters: Max_Iteration, force F, masses of agents: m, G_0, initial position of agents: X_0, inertia weights, c'_1, c'_2, d etc.
    Initialize gBest, gBestScore to large value
    while iteration (t) ≤ Max_Iteration do
        Update gravitational constant for time t: G(t)
        for each agents do
            Set weights of ANN(V, W and biases) using position of agents
            for each epoch do
                Compute error (E) of current agents
                Sum = Sum + E
            end for
            MSE = Sum/NoP
            Current Fitness of agent i: CF(i) = MSE
            if gBestScore > CF(i) then
                gBestScore = CF(i)
                gBest = X(i,:)
            end if
        end for
        for each agents do
            Update mass m
            Calculate force F exerted from other agents
            Calculate acceleration a
            Update Velocity V
            Update new position of agents X
        end for
    end while
end procedure
```

4.1 NSL-KDD dataset description

NSL-KDD dataset is an improved version of the popular KDD Cup'99 dataset. It solves some inherent problems of the KDD'99 dataset (Tavallaee et al. 2009; McHugh 2000). Due to lack of public datasets for network-based IDS, the current version of NSL-KDD may be applied as an effective benchmark dataset for this work. Furthermore, major improvements are carried out on KDD'99 to obtain NSL-KDD and this is also more advantageous over KDD'99. The number of instances in the NSL-KDD train and test sets is reasonable, which makes experimentation bias less by running experiments on whole dataset instead of running on randomly selected short portion of KDD'99 dataset. It is free of redundant records in train and test dataset, so the classifiers will not be biased towards repeated instances in the dataset. The NSL-KDD dataset contains two different files for training and testing, and hence, there is no overhead of dividing the dataset into training and testing which also makes a slight contribution towards performance evaluation of the learning techniques. Table 1 shows the properties of the NSL-KDD train and test datasets (both are obtained as .txt files). It should be noted that there is no missing values in any attribute, and number of attributes in the table includes a class attribute. One epoch during the training phase refers to processing of all the patterns or instances present in the KDDTrain+.txt dataset. However, size of testing set is 15 % of size of whole dataset. Since these datasets are large and computation in ordinary machines might take too much time or sometimes might not support the memory requirements, randomly chosen 10 % of the training dataset was used during training. The random selection of 10 % of the training dataset was repeated for different executions so that there would be comparatively less chance of repeating the training data in different simulations. There are different types of attacks in the dataset. However, in this work, it is considered as a two-class problem where patterns may belong to either 'normal' or 'anomaly' class.

Table 1 Dataset properties

Dataset	No. of records	No. of attributes
KDDTrain+.txt	125,973	42
KDDTest+.txt	22,544	42

4.2 Normalization of dataset

In the NSL-KDD datasets, the values for each attributes are often not distributed uniformly. It is wise to maintain a uniform distribution of each input attributes in the dataset before processing in the neural network. Hence, to ensure that the input values were compatible despite significant differences in their values, the dataset is normalized with respect to each input value using the formula given in Eq. (17), where d_i is the original value; $d_{\max}$ and $d_{\min}$ are the maximum and minimum value, respectively, in the input attribute from which d_i is obtained. Then, normalized value of d_i is denoted as d'_i. However, it was seen that the normalized dataset sometimes contained majority of zeros and in such cases it was preferred to use the in-built data normalization function of MATLAB called *mapminmax* which normalizes data in the range [−1,1].

$$d'_i = \frac{d_i - d_{\min}}{d_{\max} - d_{\min}} \tag{17}$$

4.3 Simulation parameters

Setting parameters for both ANN and the training algorithms does also play a great role towards performance evaluation

of the designed models. In this work, following different parameters are considered based on many previous machine learning applications. In GD training, learning rate (α) is set to 0.99; momentum factor (μ) is set to 0.50. In GA, chromosome population size is set to 50; crossover and mutation probabilities are set to 0.5 and 0.3, respectively. In PSO, particles population size is 30; inertia weight (w), c_1' and c_2' are set to 2 each for good convergence behavior; inertia weight decreasing over time from 0.9 to 0.5 which has the effect of narrowing the search, which changes gradually from exploratory to an exploitative mode, as mentioned in one of the available literature (Eberhart and Kennedym 1995). Similarly, for GS, size of population is 30, G_0 is 1 and the constant η is set to 20 (Rashedi et al. 2009; Mirjalili et al. 2012; Dash et al. 2015b). It should be noted that the parameters used in GS and PSO are also used in GSPSO- based training. As the designed ANN is a multilayered (input–hidden–output) ANN, the number of hidden units (M) is set to 17 determined using an empirical formula as given in following equation; where k is a constant set to 10 in this work. In all the five ANN techniques, number of epochs is set to 100.

$$M = \sqrt{(N+1)} + k \tag{18}$$

4.4 Obtained results

All the simulations are carried out for ten times in each technique to obtain an average result which can be best suited for comparison. From all the ten simulations, minimum, maximum, mean and standard deviation (±std) are evaluated. Tables 2 and 3 show performance of different methods during training with KDDTrain+.txt dataset where MSE and training time are taken as performance evaluation factor. Figure 2 gives fitness evolution curve for all the ANN models used in this work. Tables 4 and 5 show detection accuracy rate and detection time for different methods.

Tables 2 and 3 depict average results obtained by different methods when the models are trained KDDTrain+.txt dataset for 100 epochs when simulated for ten independent runs. It should be noted that the training time refereed to in Table 3 is the simulation time required for each epoch in different ANN models implemented in this work. Table 2 shows that the GD-ANN is not able to converge to minimum within the set epoch limit and is achieving an MSE 1.233 which is not quite well for neural network classification models; rather it is getting stuck in the local minima of the evolution curve. Other models are converging from a maximum error to a minimum within in 100 epochs which has been shown in the evolution curves given in Fig. 2. The results indicate that GSPSO-ANN also improves the capability of ANN partially to avoid local minima. For the minimum MSE over ten independent simulations, PSO-ANN, GS-ANN and GSPSO-ANN exhibit close results, but GSPSO-ANN provides the minimum of all these

Table 2 Mean squared error (MSE) obtained by different models with KDDTrain+.txt dataset

Method	MSE			
	Min	Max	Average	±std
DT	0.005	0.008	0.4533	0.012
GD-ANN	1.233	1.69	1.3608	0.1435
GA-ANN	0.1161	2.1871	0.6431	0.5512
PSO-ANN	0.062	1.98	0.5724	0.6109
GS-ANN	0.042	2.05	0.4317	0.6081
GSPSO-ANN	0.0379	2.1	0.4527	0.6175

Table 3 Time taken to train different models with KDDTrain+.txt dataset

Method	Training time in seconds			
	Min	Max	Average	±std
DT	1.79	3.50	2.36	0.5032
GD-ANN	210.67	309.46	269.31	30.8055
GA-ANN	134.00	180.00	156.85	13.9957
PSO-ANN	74.00	96.00	84.03	5.8841
GS-ANN	167.00	180.00	171.50	4.4535
GSPSO-ANN	90.20	123.00	103.70	10.7564

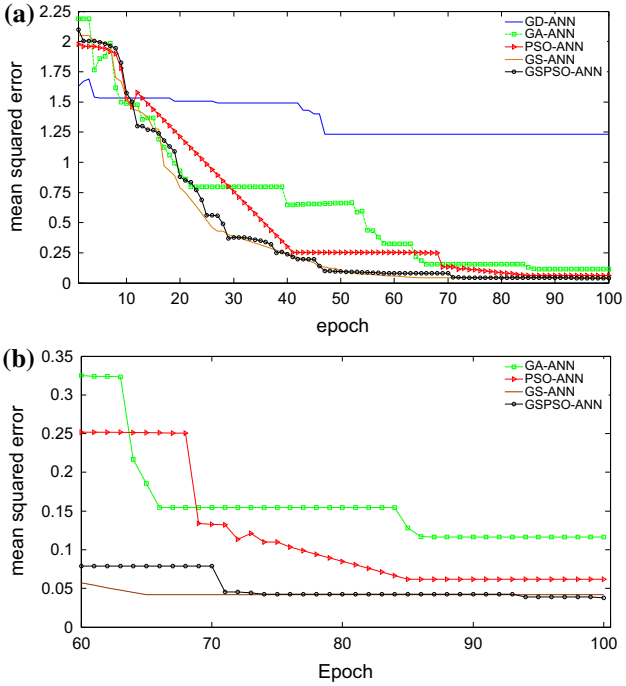

Fig. 2 Epoch wise fitness evolution of ANN models: **a** comparing all five ANN models, **b** enlarged view from 60 to 100 epochs

Table 4 Detection accuracy rate resulting from all the used models

Method	Detection rate (%)			
	Min	Max	Average	±std
DT	89.30	93.81	92.01	1.4102
GD-ANN	78.01	82.36	80.45	1.5197
GA-ANN	83.11	88.34	86.37	1.8958
PSO-ANN	88.90	94.79	92.06	1.9901
GS-ANN	89.67	94.90	92.81	2.0078
GSPSO-ANN	90.34	98.13	95.26	3.3121

Table 5 Average time taken to test a single instance of whole dataset (KDDTrain+.txt and KDDTest+.txt)

Method	Average detection time in seconds			
	Min	Max	Average	±std
DT	1.40	1.53	1.46	0.0456
GD-ANN	0.81	1.66	1.17	0.2634
GA-ANN	0.70	1.30	0.97	0.2025
PSO-ANN	0.90	1.21	1.08	0.0930
GS-ANN	0.90	1.35	1.17	0.1390
GSPSO-ANN	0.83	1.45	1.03	0.1741

three results with a slightly more expenditure of computation time.

Table 4 shows that DT is giving the better results as compared with other techniques and it does not need too much computations while generating a decision tree. In this work, maximum time taken by DT is 3.5 seconds which is negligible when compared with 310 seconds taken by GD-ANN. The GS-ANN is taking more time to learn the training patterns as compared with PSO-ANN or GA-ANN, whereas GSPSO-ANN is better than GS-ANN whose training time lies in between that of PSO-ANN and GS-ANN.

Testing results shown in Table 4 show that GS-ANN and GSPSO-ANN are better than DT, and other three ANN approaches. From the table, it can be inferred that GSPSO-ANN has a better detection rate than other ANN-based methods used in this work. With regard to the maximum detection rate resulting from all methods, GS-ANN and GSPSO-ANN have 94.90 and 98.13 % detection accuracy, which are better than that obtained by other four methods. Based on the results reported in Table 4 with corresponding testing time in Table 5, it can be revealed that the GSPSO-ANN approach outperforms other methods. The GSPSO-ANN method is also reliable because it is reducing the probability of getting trapped in the local minima. However, the GS-ANN is not satisfactory to report better

Table 6 p values reported by Wilcoxon's rank sum test comparing GS-ANN and GSPSO-ANN with other methods

Versus→	DT	GD-ANN	GA-ANN	PSO-ANN	GS-ANN
GS-ANN	0.2111	0.000176	0.000174	0.2401	–
GSPSO-ANN	0.063	0.000173	0.000171	0.0306	0.043

detection rate because of its slow searching process which affects exploitation ability of the ANN.

Generally, in all results reported after the training and testing of the different techniques, it could be observed that GS-ANN does not give a better performance as compared to the GSPSO-ANN because of the slow searching process of the GS algorithm, which affects GS-ANNs exploitation ability. However, GS has strong exploration ability among many different evolutionary algorithms such as GA and PSO (Rashedi et al. 2009). Learning algorithms for ANNs need not only strong exploration ability but also precise exploitation ability and hence GSPSO could be a wise choice to be applied to the IDS research. Referring to the results of the detection accuracy obtained by FNN- based algorithms, it could be seen that GS-ANN performs better than PSO-ANN, GA-ANN, GD-ANN and DT. However, both these algorithms (GS, PSO, GA, and GD) still suffer from the problem of getting trapped in the local minima. This weakness of both the methods gives a comparatively unstable performance as compared to GSPSO-ANN.

A non-parametric statistical significance proof known as the Wilcoxon's rank sum test for independent samples (Garcia et al. 2009) with a signification level of 5 % has been conducted over the accuracy results obtained in 10 simulations. Table 6 reports the p values produced by this test for the pair-wise comparison of the detection accuracy of different groups, viz. GS-ANN versus DT, GS-ANN versus GD-ANN, etc. As per null hypothesis, it is assumed that there is no significant difference between mean values of the two considered algorithms. However, an alternative hypothesis considers a significant difference between accuracy values in both approaches provides a better confidence level, e.g., if the significance level is 5 % (=0.05), then the confidence level is 95 % which is a strong evidence against the null hypothesis. Therefore, such an evidence will indicate that results obtained by GS-ANN and GSPSO-ANN are statistically significant and have not occurred by coincidence, i.e., due to common noise contained during testing. The tabulated statistical results reveal that both GS-ANN and GSPSO-ANN models could demonstrate good statistical significance with regard to their counterparts for the tested NSL-KDD dataset. However, the p value obtained for GS-ANN against DT and PSO-ANN is greater than the significance level of 5 %, which means that the GS-ANN method fails to reject the null hypothesis and hence it is not 95 %

Table 7 Information on class distribution in MAWI datasets

Dataset file	$\mathcal{N}_{\text{instances}}$	$\mathcal{N}_{\text{attr}}$	$\mathcal{N}_{\text{ssh}}$	$\mathcal{N}_{\text{notssh}}$	$\%_{\text{ssh}}$	$\%_{\text{notssh}}$
MAWI-1.txt	500,000	23	15	499,985	0.0030	99.9970
MAWI-2.txt	500,001	23	4	499,997	0.0008	99.9992
MAWI-3.txt	500,001	23	7	499,994	0.0014	99.9986
MAWI-4.txt	500,001	23	8	499,993	0.0016	99.9984
MAWI-5.txt	500,001	23	6	499,995	0.0012	99.9988
MAWI-6.txt	500,001	23	7	499,994	0.0014	99.9986
MAWI-7.txt	500,001	23	15	499,986	0.0030	99.9970
MAWI-8.txt	500,001	23	6	499,995	0.0012	99.9988
MAWI-9.txt	500,001	23	3	499,998	0.0006	99.9994
MAWI-10.txt	500,001	23	7	499,994	0.0014	99.9986
MAWI-11.txt	500,001	23	9	499,992	0.0018	99.9982
MAWI-12.txt	500,001	23	9	499,992	0.0018	99.9982
MAWI-13.txt	500,001	23	237	499,764	0.0474	99.9526
MAWI-14.txt	500,001	23	565	499,436	0.1130	99.8870
MAWI-15.txt	500,001	23	700	499,301	0.1400	99.8600
MAWI-16.txt	500,001	23	361	499,640	0.0722	99.9278
MAWI-17.txt	500,001	23	91	499,910	0.0182	99.9818
MAWI-18.txt	500,001	23	40	499,961	0.0080	99.9920
MAWI-19.txt	500,001	23	37	499,964	0.0074	99.9926
MAWI-20.txt	500,001	23	36	499,965	0.0072	99.9928
MAWI-21.txt	500,001	23	1973	498,028	0.3946	99.6054
MAWI-22.txt	500,001	23	4088	495,913	0.8176	99.1824
MAWI-23.txt	500,001	23	2311	497,690	0.4622	99.5378
MAWI-24.txt	500,001	23	1519	498,482	0.3038	99.6962
MAWI-25.txt	500,001	23	1095	498,906	0.2190	99.7810
MAWI-26.txt	500,001	23	960	499,041	0.1920	99.8080
MAWI-27.txt	500,001	23	752	499,249	0.1504	99.8496
MAWI-28.txt	500,001	23	625	499,376	0.1250	99.8750
MAWI-29.txt	500,001	23	576	499,425	0.1152	99.8848
MAWI-30.txt	500,001	23	497	499,504	0.0994	99.9006
MAWI-31.txt	500,001	23	380	499,621	0.0760	99.9240
MAWI-32.txt	500,001	23	358	499,643	0.0716	99.9284
MAWI-33.txt	500,001	23	378	499,623	0.0756	99.9244
MAWI-34.txt	500,001	23	262	499,739	0.0524	99.9476
MAWI-35.txt	500,001	23	265	499,736	0.0530	99.9470
MAWI-36.txt	500,001	23	238	499,763	0.0476	99.9524
MAWI-37.txt	500,001	23	163	499,838	0.0326	99.9674
MAWI-38.txt	500,001	23	188	499,813	0.0376	99.9624
MAWI-39.txt	500,001	23	140	499,861	0.0280	99.9720
MAWI-40.txt	473,803	23	85	473,718	0.0179	99.9821

confident of producing better results than these two methods (DT and PSO-ANN) in future. However, GSPSO-ANN outperforms other ANN-based techniques in terms of statistical significance. However, the results obtained against DT are statistically insignificant where the p value is reported to be 0.063.

4.5 Testing the applicability of proposed methods in other dataset

To test the applicability of proposed methods in future, a different dataset is used which is known as 'MAWI working group test archieve' or 'MAWI' (Alshammari and Zincir-

Heywood 2007). MAWI is basically an archive containing SSH traffic information resulting from requests from different applications. There are two different types of traffic information available in MAWI dataset namely, 'SSH' and 'NOTSSH'. The interesting fact about this dataset is that it is too much imbalanced dataset where SSH type traffic instances are very rarely available and a maximum of 1 % chance of occurrence in whole dataset. Hence, it becomes a challenge for any machine learning- based predictor. There are a total of 40 files containing the SSH traffic information, which are briefly depicted in Table 7. More information on this dataset is provided in online supplementary resource.

In Table 7, $\mathcal{N}_{\text{instances}}$ denotes total number of instances in the dataset, $\mathcal{N}_{\text{attr}}$ denotes the number of attributes including the class attribute, $\mathcal{N}_{\text{ssh}}$ and $\%_{\text{ssh}}$ denote number of instances of SSH class and their distribution percentage in whole dataset, respectively. Similarly, $\mathcal{N}_{\text{notssh}}$ and $\%_{\text{notssh}}$ denote number of instances of NOTSSH class and their distribution percentage in whole dataset, respectively. As already mentioned, one can see that the SSH type instances are too rare and occur by chance in all the 40 datasets of the MAWI repository. It should be noted that similar data normalization procedure has been followed as it was carried out for NSL-KDD dataset. As, it is a 2-class problem, the target class label 'NOTSSH' in the dataset was replaced by 0 and the class label 'SSH' was replaced by 1. As the number of input attributes is 22 (i.e., 23 − 1), number of hidden nodes was set to 15 which was found out using Eq. (18), i.e., $M = \sqrt{(N = 22) + 1} + (k = 10) = \lceil 14.79 \rceil$.

In this work, only the GS-ANN and GSPSO-ANN models are tested on all these MAWI datasets and results are reported. It is worth mentioning that all the simulations (preprocessing, training and testing) are carried out in a similar system as in NSL-KDD with Quad-Core CPU and 4 GB RAM. All other simulation parameters in GS-ANN and GSPSO-ANN are set to same values as they were set for NSL-KDD dataset.

It is well known that neural networks are prone to data over-fitting when trained with highly imbalanced distributions of different classes. As the MAWI datasets are too large and highly imbalanced, it is not wise to train the models with large training set which may also increase unnecessary computation overhead. In this work, as a very high number of NOTSSH type instances are present, the training patterns were selected for the training dataset in SSH to NOTSSH type distribution proportion of 1:2. More clearly, the training set was prepared in the following two steps:

- Include all the SSH type instances from the dataset in the training set (total $\mathcal{N}_{\text{ssh}}$ number of instances).
- Randomly select $2 \times \mathcal{N}_{\text{ssh}}$ instances of NOTSSH type instances from dataset and include them in the training set.

Table 8 Performance of GS-ANN for MAWI datasets

Dataset	MMSE	$\mathcal{T}_{\text{train}}$	Accuracy (%)	$\mathcal{T}_{\text{test}}$
MAWI-1.txt	0.00	0.20	0.00	≈0.002
MAWI-2.txt	0.04	0.07	99.99	
MAWI-3.txt	0.23	0.10	99.99	
MAWI-4.txt	0.17	0.12	99.99	
MAWI-5.txt	0.13	0.09	99.99	
MAWI-6.txt	0.19	0.10	99.99	
MAWI-7.txt	0.23	0.20	99.99	
MAWI-8.txt	0.21	0.09	99.99	
MAWI-9.txt	0.00	0.06	1.13	
MAWI-10.txt	0.14	0.11	99.99	
MAWI-11.txt	0.11	0.13	99.99	
MAWI-12.txt	0.23	0.13	99.99	
MAWI-13.txt	0.05	2.86	72.07	
MAWI-14.txt	0.11	6.77	64.55	
MAWI-15.txt	0.05	8.39	0.14	
MAWI-16.txt	0.08	4.33	33.34	
MAWI-17.txt	0.05	1.11	99.98	
MAWI-18.txt	0.11	0.50	5.95	
MAWI-19.txt	0.13	0.46	0.00	
MAWI-20.txt	0.14	0.45	99.99	
MAWI-21.txt	0.01	23.64	99.60	
MAWI-22.txt	0.08	49.04	99.18	
MAWI-23.txt	0.03	34.20	0.46	
MAWI-24.txt	0.11	22.56	71.59	
MAWI-25.txt	0.03	13.16	3.15	
MAWI-26.txt	0.03	11.52	0.19	
MAWI-27.txt	0.03	8.86	69.14	
MAWI-28.txt	0.09	7.58	4.82	
MAWI-29.txt	0.10	6.76	99.88	
MAWI-30.txt	0.04	5.84	98.75	
MAWI-31.txt	0.05	4.48	0.07	
MAWI-32.txt	0.12	5.42	99.92	
MAWI-33.txt	0.10	6.86	99.92	
MAWI-34.txt	0.04	4.19	99.94	
MAWI-35.txt	0.04	3.89	99.94	
MAWI-36.txt	0.04	3.08	98.24	
MAWI-37.txt	0.08	2.12	99.96	
MAWI-38.txt	0.08	2.38	0.03	
MAWI-39.txt	0.09	1.66	63.38	
MAWI-40.txt	0.11	1.02	0.01	

Hence, the total number of instances in the training dataset is $3 \times \mathcal{N}_{\text{ssh}}$. For all the 40 MAWI datasets, the above-mentioned steps were followed before the training of the considered ANN models (GS-ANN, GSPSO-ANN). However, after the training phase, whole dataset was used for testing the trained ANN model. To evaluate the model, four

Table 9 Performance of GSPSO-ANN for MAWI datasets

Dataset	MMSE	$\mathcal{T}_{\text{train}}$	Accuracy (%)	$\mathcal{T}_{\text{test}}$
MAWI-1.txt	0.00	0.20	0.00	≈0.002
MAWI-2.txt	0.00	0.07	64.33	
MAWI-3.txt	0.23	0.10	63.44	
MAWI-4.txt	0.13	0.12	70.99	
MAWI-5.txt	0.07	0.09	32.39	
MAWI-6.txt	0.18	0.10	64.94	
MAWI-7.txt	0.22	0.20	99.99	
MAWI-8.txt	0.06	0.09	99.15	
MAWI-9.txt	0.00	0.06	0.00	
MAWI-10.txt	0.18	0.10	94.00	
MAWI-11.txt	0.15	0.13	65.70	
MAWI-12.txt	0.18	0.13	62.15	
MAWI-13.txt	0.02	2.78	65.76	
MAWI-14.txt	0.03	6.62	99.35	
MAWI-15.txt	0.10	8.19	96.76	
MAWI-16.txt	0.01	4.22	99.32	
MAWI-17.txt	0.08	1.09	65.87	
MAWI-18.txt	0.09	0.49	92.98	
MAWI-19.txt	0.13	0.45	66.99	
MAWI-20.txt	0.10	0.44	94.67	
MAWI-21.txt	0.01	23.04	68.19	
MAWI-22.txt	0.05	55.92	81.41	
MAWI-23.txt	0.02	34.60	99.44	
MAWI-24.txt	0.05	18.76	70.86	
MAWI-25.txt	0.02	13.03	69.80	
MAWI-26.txt	0.04	11.91	95.43	
MAWI-27.txt	0.03	9.14	95.12	
MAWI-28.txt	0.03	7.57	69.07	
MAWI-29.txt	0.05	7.01	68.49	
MAWI-30.txt	0.02	6.19	69.35	
MAWI-31.txt	0.04	4.72	93.64	
MAWI-32.txt	0.04	4.42	68.14	
MAWI-33.txt	0.02	4.63	69.48	
MAWI-34.txt	0.03	4.10	67.03	
MAWI-35.txt	0.02	3.18	64.98	
MAWI-36.txt	0.02	2.84	98.11	
MAWI-37.txt	0.13	1.96	62.36	
MAWI-38.txt	0.04	2.26	62.45	
MAWI-39.txt	0.04	1.68	97.42	
MAWI-40.txt	0.06	1.03	61.82	

parameters are considered such as minimum MSE (denoted as MMSE), training time per epoch in seconds (denoted as $\mathcal{T}_{\text{train}}$), detection accuracy in %, and testing time per instance in seconds (denoted as $\mathcal{T}_{\text{test}}$).

Tables 8 and 9 depict all the results obtained after applicability test of GS-ANN and GSPSO-ANN, respectively, for MAWI dataset. As all the 40 MAWI datasets are independent of each other hence there is no correlation of results obtained for each datasets. It can be observed that the GS-ANN has performed worse in terms of accuracy as compared to its counterpart GSPSO-ANN. And, there is a higher deviation in results obtained by GS-ANN as compared to that of the GSPSO-ANN. The possible reasons behind this type of performance of GS-ANN could be either under-fitting or over-fitting where it could not capture the training set well enough or could not fit the training data. However, the GSPSO-ANN has performed better in terms of training data fitting by the model. To study the over performance of both these models, statistical results are obtained which have been depicted in Table 10. From the table, it is seen that GSPSO-ANN outperformed GS-ANN while achieving a better minimum error (MMSE) as well as better accuracy of detection. However, the p value obtained from the ranksum test between the GS-ANN and GSPSO-ANN model is 0.116133 which could not able to reject the null hypothesis at a significance level of 5 %. But, it rejects the null hypothesis at a higher significance level of 11.61 % which may not be a strong confidence for machine learning models.

Table 11 depicts a comparison of detection accuracy achieved by some of the intrusion detection models used for designing IDS which are available in the literature.

5 Conclusions and future directions

Two new intrusion detection approaches (IDS) have been proposed: GS-ANN and GSPSO-ANN based on evolutionary algorithms such as GS and PSO. The applicability of these proposed approaches with regard to mean squared error, training time, decision time and detection accuracy rate have been obtained by testing with NSL-KDD dataset and the results have also been compared with popularly used techniques such as DT, GD-ANN, GA-ANN and PSO-ANN. It has been found that the GS-ANN is taking more time to learn the training patterns as compared with the PSO-ANN or the GA-ANN, whereas GSPSO-ANN is better than the GS-ANN whose training time lies in between that of the PSO-ANN and the GS-ANN. With regard to the maximum detection rate resulting from all methods, GS-ANN and GSPSO-ANN have 94.90 and 98.13 % detection accuracy, which are better than that obtained by other four different methods when tested with NSL-KDD dataset. It has also been found that the results obtained by GS-ANN and GSPSO-ANN are statistically significant based on Wilcoxon's rank sum test as compared to other techniques considered in this work. It has also been seen that the two proposed models, GS-ANN and GSPSO-ANN could also perform well in case of a highly imbalanced dataset called MAWI dataset. However, this work is completely based on existing intrusion detection datasets

Table 10 Comparison of overall performance of GS-ANN and GSPSO-ANN for MAWI dataset

Model	MMSE (mean±std)	$\mathcal{T}_{\text{train}}$ (mean±std)	Accuracy (%) (mean±std)	p value
GS-ANN	0.0925 ± 0.0634	6.11 ± 10.08	64.63 ± 44.26	0.116133
GSPSO-ANN	0.0680 ± 0.0631	6.09 ± 10.75	73.28 ± 23.48	

Table 11 Comparisons of proposed models with other works by best or average detection rate

Author/reference	Dataset used	Best/average detection rate (%)
Sindhu et al. (2009)	KDD Cup	90.16
Levin (2000)	KDD Cup	91.5
Pfahringer (2000)	KDD Cup	91.8
Toosi and Kahani (2007)	KDD Cup	95.3
Horng et al. (2011)	KDD Cup	95.7
Wang et al. (2010)	KDD Cup	96.71
Ahmad et al. (2014)	KDD Cup	99.6
Kim et al. (2014)	NSL-KDD	99.0 (ROC curve)
GS-ANN (this work)	NSL-KDD	94.9 (best), 92.81 (average)
GSPSO-ANN (this work)	NSL-KDD	98.13 (best), 95.26 (average)
GS-ANN (this work)	MAWI	64.63 (average)
GSPSO-ANN (this work)	MAWI	73.28 (average)

and an appropriate feature selection mechanism has not been chosen. Therefore, in future, it would be possible to apply a good feature selection strategy or apply some other neural network model for efficient detection of intrusion.

Acknowledgments The author expresses his sincere gratitude to Dr. Prabhat K. Sahu (NIST Berhampur) and Dr. Sukanta Mondal (BITS-Pilani, Goa) for their fruitful suggestions during the work. The author thankfully acknowledges NIST Berhampur and BITS-Pilani, Goa for providing required support. The author thanks anonymous reviewers and the editor for their valuable comments and suggestions to improve the quality of the paper.

Compliance with ethical standards

Conflict of interest The author declares that there is no conflict of interest regarding publication of this work.

Appendix

Complexity analysis of proposed model

As the proposed ANN models are three-layered architecture, i.e., (input–hidden–output), the computation will be carried out in two phases; (i) forward computation is from input to hidden layer which occurs in $O(NM)$ times, (ii) the next computation is from hidden to output layer which takes $O(M)$ times. So total is $T(N, M) = O(NM) + O(M) \approx O(NM)$. For proposed GS-ANN and GSPSO-ANN, $O(NM)$ computations will be carried out for each agent in the system. Hence, the overall algorithmic time complexity of the proposed models is $O(NM)$ multiplied with population size (N_{pop}). Therefore, algorithmic time complexity of the model is $O(N_{\text{pop}} \times NM)$.

References

Ahmad I, Hussain M, Alghamdi A, Alelaiwi A (2014) Enhancing SVM performance in intrusion detection using optimal feature subset selection based on genetic principal components. Neural Comput Appl 24(7–8):1671–1682

Alshammari R, Zincir-Heywood AN (2007) A flow based approach for SSH traffic detection. IEEE Int Conf Syst Man Cybern 2007:296–301

Aydin MA, Zaim AH, Ceylan KG (2009) A hybrid intrusion detection system design for computer network security. Comput Electr Eng 35:517–526

Bao F, Chen I-R, Chang M, Cho J-H (2012) Hierarchical trust management for wireless sensor networks and its applications to trust-based routing and intrusion detection. IEEE Trans Netw Serv Manag 9(2):169–183

Beghdad R (2008) Critical study of neural networks in detecting intrusions. Comput Secur 27(5–6):168–175

Catania CA, Garino CG (2012) Automatic network intrusion detection: current techniques and open issues. Comput Electr Eng 38:1062–1072

Chen YH, Abraham A, Yang B (2007) Hybrid flexible neural-tree-based intrusion detection systems. Int J Intell Syst 22(4):337–352

Dainotti A, Pescapé A, Rossi PS, Palmieri F, Ventre G (2008) Internet traffic modeling by means of hidden Markov models. Comput Netw 52(14):2645–2662

Dash T, Nayak SK, Behera HS (2015a) Hybrid gravitational search and particle swarm based fuzzy MLP for medical data classification. In: Computational intelligence in data mining, vol 1. Springer, India, pp 35–43

Dash T, Nayak T, Swain RR (2015b) Controlling wall following robot navigation based on gravitational search and feed forward neural

network. In: Proceedings of the 2nd international conference on perception and machine intelligence, ACM, pp 196–200
Eberhart R, Kennedym J (1995) A new optimization using particle swarm theory. In: Sixth international symposium on micro machine and human science, MHS'95, IEEE, pp 39–43
Fiore U, Palmieri F, Castiglione A, De Santis A (2013) Network anomaly detection with the restricted Boltzmann machine. Neurocomputing 122:13–23
Garcia S, Molina D, Lozano M, Herrera F (2009) A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 special session on real parameter optimization. J Heuristics 15:617–644
Gómez J, Gil C, Baños R, Márquez AL, Montoya FG, Montoya MG (2013) A Pareto-based multi-objective evolutionary algorithm for automatic rule generation in network intrusion detection systems. Soft Comput 17(2):255–263
Gu Y, McCallum A, Towsley D (2005) Detecting anomalies in network traffic using maximum entropy estimation, In: Proceedings of the 5th ACM SIGCOMM conference on internet measurement, IMC'05. USENIX Association, Berkeley, CA, USA, p 32
Horng S-J, Su M-Y, Chen Y-H, Kao T-W, Chen R-J, Lai J-L, Perkasa CD (2011) A novel intrusion detection system based on hierarchical clustering and support vector machines. Expert Syst Appl 38:306–313
Hu J, Yu X, Qiu D (2009) A simple and efficient hidden Markov model scheme for host-based anomaly intrusion detection. IEEE Netw 23:42–47
KDD Cup (1999) Intrusion detection data set. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
Kim G, Lee S, Kim S (2014) A novel hybrid intrusion detection method integrating anomaly detection with misuse detection. Expert Syst Appl 41(4):1690–1700
Kuang F, Zhang S, Jin Z, Xu W (2015) A novel SVM by combining kernel principal component analysis and improved chaotic particle swarm optimization for intrusion detection. Soft Comput 19:1187–1199
Kumar PAR, Selvakumar S (2011) Distributed denial of service attack detection using an ensemble of neural classifier. Comput Commun 34:1328–1341
Lazarevic A, Kumar V, Srivastava J (2005) Intrusion detection: a survey. In: Managing cyber threats. Massive Computing, vol 5. Springer, New York, pp 19–78
Lee W, Nimbalkar RA, Yee KK, Patil SB, Desai PH, Tran TT, Stolfo SJ (2000) A data mining and CIDF based approach for detecting novel and distributed intrusions. In: Recent advances in intrusion detection, Springer, Berlin, pp 49–65
Levin I (2000) KDD-99 classifier learning contest LLSoft's results overview. SIGKDD Explor 1(2):67–75
Linda O, Vollmer T, Manic M (2009) Neural network based intrusion detection system for critical infrastructures. In: Proceedings of international joint conference on neural networks, Atlanta, Georgia, USA, 14–19 June 2009, pp 1827–1834
Mabu S, Chen C, Lu N, Shimada K, Hirasawa K (2011) An intrusion-detection model based on fuzzy class-association-rule mining using genetic network programming. IEEE Trans Syst Man Cyber Part C 41(1):130–139
Mahoney MV, Chan PK (2002) Learning nonstationary models of normal network traffic for detecting novel attacks. In: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining. KDD02. ACM, New York, NY, USA, pp 376-385
Manikopoulos C, Papavassiliou S (2002) Network intrusion and fault detection: a statistical anomaly approach. IEEE Commun Mag 40(10):76–82
Marín-Blázquez JG, Pérez GM (2009) Intrusion detection using a linguistic hedged fuzzy-XCS classifier system. Soft Comput 13(3):273–290
McHugh J (2000) Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory. ACM Trans Inf Syst Secur 3(4):262–294
Mirjalili S, Hashim SZM, Sardroudi HM (2012) Training feedforward neural networks using hybrid particle swarm optimization and gravitational search algorithm. Appl Math Comput 218:11125–11137
Palmieri F, Fiore U, Castiglione A (2014) A distributed approach to network anomaly detection based on independent component analysis. Concurr Comput 26(5):1113–1129
Palmieri F, Fiore U (2010) Network anomaly detection through nonlinear analysis. Comput Secur 29(7):737–755
Patcha A, Park JM (2007) An overview of anomaly detection techniques: existing solutions and latest technological trends. Comput Netw 51(12):3448–3470
Pfahringer B (2000) Winning the KDD99 classification cup: bagged boosting. SIGKDD Explor 1(2):65–66
Ramasubramanian P, Kannan A (2006) A genetic-algorithm based neural network short-term forecasting framework for database intrusion prediction system. Soft Comput 10(8):699–714
Rashedi E, Nezamabadi-pour H, Saryazdi S (2009) GSA: a gravitational search algorithm. Inf Sci 179:2232–2248
Rastegari S, Hingston P, Lam CP (2015) Evolving statistical rulesets for network intrusion detection. Appl Soft Comput 33:348–359
Shakshuki EM, Kang N, Sheltami TR (2013) EAACKA secure intrusion-detection system for MANETs. IEEE Trans Ind Electron 60(3):1089–1098
Shin S, Kwon T, Jo G-Y, Park Y, Rhy H (2010) An experimental study of hierarchical intrusion detection for wireless industrial sensor networks. IEEE Trans Ind Inf 6(4):744–757
Sindhu SSS, Geetha S, Marikannan M, Kannan A (2009) A neuro-genetic based short-term forecasting framework for network intrusion prediction system. Int J Autom Comput 6(4):406–414
Tavallaee M, Bagheri E, Lu W, Ghorbani A (2009) A detailed analysis of the KDD CUP'99 dataset. In: Proceedings of the IEEE symposium on computational intelligence for security and defense applications, pp 53–58
Toosi AN, Kahani M (2007) A new approach to intrusion detection based on an evolutionary soft computing model using neuro-fuzzy classifiers. Comput Commun 30:2201–2212
Vollmer T, Alves-Foss J, Manic M (2011) Autonomous rule creation for intrusion detection. In: IEEE symposium on computational intelligence in cyber security (CICS), pp 1–8
Wang G, Hao J, Ma J, Huang L (2010) A new approach to intrusion detection using artificial neural networks and fuzzy clustering. Expert Syst Appl 37:6225–6232
Wu SX, Banzhaf W (2008) The use of computational intelligence in intrusion detection systems: a review. Technical report #2008-05, Memorial University of Newfoundland
Wu S, Yen E (2009) Data mining-based intrusion detectors. Expert Syst Appl 36(3):5605–5612
Ye N, Emran SM, Chen Q, Vilbert S (2002) Multivariate statistical analysis of audit trails for host-based intrusion detection. IEEE Trans Comput 51(7):810820
Yu Z, Tsai JJP, Weigert T (2007) An automatically tuning intrusion detection system. IEEE Trans Syst Man Cybern Part B 37(2):373–384
Zbilut JP, Webber CL (2006) Recurrence quantification analysis. In: Akay M (ed) Wiley encyclopedia of biomedical engineering. Wiley, Hoboken