IEREK PRESS
Academic Research Community
http://www.press.ierek.com
ARChive
ISSN (Print: 2537-0154, online: 2537-0162)
International Journal on:
The Academic Research Community Publication

DOI: 10.21625/archive.v2i4.397

A Proposed Heuristic Optimization Algorithm for Detecting Network Attacks

Amr Hassan Yassin[1], Hany Hamdy Hussien[2]

[1] *1Lecturer, Electronics & Communication, Engineering Department, Alexandria Higher Institute of Engineering and Technology*

[2] *Lecturer, Higher Institute of Management Science and Advanced Computing, El-Beheira, Egypt.*

Abstract

Due to the exponential growth of E-Business and computing capabilities over the web for a pay-for-use groundwork, the risk factors regarding security issues also increase rapidly. As the usage increases, it becomes very difficult to identify malicious attacks since the attack patterns change. Therefore, host machines in the network must continually be monitored for intrusions since they are the final endpoint of any network. The purpose of this work is to introduce a generalized neural network model that has the ability to detect network intrusions. Two recent heuristic algorithms inspired by the behavior of natural phenomena, namely, the particle swarm optimization (PSO) and gravitational search (GSA) algorithms are introduced. These algorithms are combined together to train a feed forward neural network (FNN) for the purpose of utilizing the effectiveness of these algorithms to reduce the problems of getting stuck in local minima and the time-consuming convergence rate. Dimension reduction focuses on using information obtained from NSL-KDD Cup 99 data set for the selection of some features to discover the type of attacks. Detecting the network attacks and the performance of the proposed model are evaluated under different patterns of network data.

© 2019 The Authors. Published by IEREK press. This is an open access article under the CC BY license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The field of computer security has turned into an extremely important issue for computer network systems through the fast growing of the computer networks over the web which might be under risk for unauthorized and intrusion process. Intrusion detection is the procedure to perceive all security issues through watching and breaking down the occasions emerging in a PC or system framework and can give a react to the unapproved activities (Yu, Wang, & Xi, 2008). Securing networks from interruptions or assaults is turning out to be more troublesome as the system innovations are expanding rapidly.

The quantity of distributed denial of service (DDoS) attacks has expanded by 90% , while the normal attacks span expanded by 28% (Wahba, El Salamouny, & El Taweel, 2015).Associations frequently convey a firewall as a first line of resistance keeping in mind the end goal to shield their private system from malicious attacks, however there are a few approaches to sidestep the firewall which makes Intrusion detection framework a moment line of barrier and an approach to screen the system activity for any conceivable danger or unlawful activity.

The performance of intrusion detection system (IDS) can be improved by using the feature selection methods,

which become essential step through reducing the list of features. Feature selection can effect on the most classification algorithms by reducing the learning time, and improving the prescient accuracy result (Wahba, El Salamouny, & El Taweel, 2015).
The remainder of this work is structured as follows. Part 2 surveys previous work about intrusion detection methods. Part 3 describes the NSL-KDD dataset. Part 4 summaries the feature selection methods. The classification phase is described in part 5. The introduced algorithm for IDS is presented in part 6. Part 7 reports the experimental results and finally the conclusion is presented in the last section.

2. Related Work

Classification, clustering and regression techniques have been used for intrusion detection systems. Several classification and clustering-based techniques have been studied for the design of intrusion detection system (IDS).

A Multilayer perceptron (MLP) and support vector machine was used in (Tang & Cao, 2009) as different approach for intrusion detection. The Average detection rate for various attacks obtain that the SVM is better than NN. They used KDD CUP'99 data set for their experiments.

Distributed Time Delay artificial neural Network (DTDNN) was selected in (Ibrahim, 2010) as classification techniques for Intrusion Detection. They used KDD CUP'99 data set for their experiments. The classification rate was high accuracy. For intrusion detection, authors use Multi-Layer Neural Networks and Naïve Bayes as classification algorithm in (Ibrahim, Badr, & Shaheen, 2012). They used gain ratio as features selection algorithms for select best features in each layer.

A three-layer and Naïve Bayes classifier was used in (Sharma & Mukherjee, 2012) as a classification algorithm for enhance the awareness of the intrusion detection through using the domain knowledge and the Backward Sequential Elimination (BSE) as feature selection technique. They used NSL-KDD dataset for their experiments.

In (Wahba, El Salamouny, & El Taweel, 2015) authors proposed an improved approach for intrusion detection based on Correlation-based Feature Selection and Information Gain hybrid as feature selection methods and naïve Bayes as the classification algorithm using reduced NSL-KDD dataset.

A hybrid multilevel network was selected in (Majeed, Hashem, & Gbashi, 2013) for intrusion detection system. The system can detect both misuse and anomaly types through using support vector machine and ANN as a multilevel classification methods based on Principle Component Analysis (PCA) as feature selection algorithm on NSL-KDD data set.

In (Ahmad, Abdulah, Alghamdi, Alnfajan, & Hussain, 2015) authors proposed a multilayer perceptron (MLP) for classification the intrusion detection system using Genetic Algorithm (GA) for features selection on DD CUP'99 data set. The method is able to reduce the feature numbers and score high detection rate, which act as prime intrusion detection mechanism.

A combination algorithm used the Bayes Net, Naive Bayes, Random Tree, PART, C4.5, and Attribute Selected Classifier in (Parvat & Chandra, 2015) as a Machine Learning algorithm. Used 20% KDD CUP'99 data set for their experiments. The purposed work improved the more accuracy as well as the time complexity.

A simple classification algorithm (nearest neighbor classifier) and feature selection technique (ant colony optimization) was used in (Aghdam & Kabiri, 2016) for intrusion detection system. The general experimental results on the KDD Cup 99 and NSL-KDD data sets providing higher accuracy in detecting intrusion attempts and lower false alarm with reduced number of features.

A new hybrid algorithm for classification and features selection was proposed in (Sadek, Soliman & Elsayed, 2013). The neural network based on indicator variable used as classification method, while rough set method used as feature selection. The experiments result on NSL-KDD dataset provided high detection accuracy with low false alarm rate as better and robust performance.

3. Dataset Description

The KDD cup 99 dataset has many disadvantages (Kaushik & Deshmukh, 2011) which discovered through different statistical analyses. NSL-KDD data set (Brunswick, 2017) is an enhanced version of the KDD data set which contains the main recodes of the all KDD data set. NSL-KDD dataset has many advantages :
The redundant records are removed from the train set to enable the classifiers to not produce any unfair result.
The duplicated records are removed to produce better reduction rates.
The records have a sufficient number in the train and test data sets to execute experiments on the complete set.
Selection procedure of records from each group in the original KDD data set is characterized by the number of these records. In each record there isa number of attributes (here 41) indicates different features of the network flow and a label assigned to each either can be classified into an attack type or as normal. The details of the NSL-KDD dataset are listed in the Table 1, 2, 3, and 4.

Table 1. Essential Features of each network connection

Att. No.	Attribute Name	Description	Sample data
1	Duration	Time connection duration	0
2	Protocol type	Protocol used in connection	TCP
3	Service	Destination network services	FTP_data
4	Flag	Status of the connection	SF
5	Src bytes	Amount of data transferred from source to destination	491
6	Dst bytes	Amount of data transferred from destination to source	0
7	Land	Status 1(source, destination, port equal) 0 (else)	0
8	Wrong fragment	Wrong fragment number	0
9	Urgent	Urgent packet numbers (bit activated)	0

Table 2. Content related features of each network connection vector

Att. No.	Attribute Name	Description	Sample data
10	Hot	Enter system & execute program	0
11	Num failed logins	Attempts login filed number	0
12	Logged in	Status 1(Success) 0 (failed)	0
13	Num compromised	No. of negotiated conditions	0
14	Root shell	Status 1(done) 0 (failed)	0
15	Su attempted	Status 1(done) 0 (failed)	0
16	Num root	No. of operations in connection	0
17	Num file creations	No. of file creation in connection	0
18	Num shells	No. of shell prompts	0
19	Num access files	No. of operations access control	0
20	Num outbound cmds	No. of outbound command in ftp session	0

Continued on next page

Table 2 continued

21	Is hot login	Status 1(hot login) 0 (otherwise)	0
22	Is guest login	Status 1(guest login) 0 (otherwise)	0

Table 3. Network connection time traffic features

Att. No.	Attribute Name	Description	Sample data
23	Count	Amount of connections (same destination)	2
24	Srv count	Amount of connections (same services + same port)	2
25	Serror rate	Percentage of connections (active flag) in count	0
26	Srv serror rate	Percentage of connections (active flag 4) in Srv count	0
27	Rerror rate	Percentage of connections (active flag) in count	0
28	Srv rerror rate	Percentage of connections (active flag 4) in Srv count	0
29	Same srv rate	Percentage of connections (same services)	1
30	Diff srv rate	Percentage of connections (different services)	0
31	Srv diff host rate	Percentage of connections (different destination)	0

Table 4. Network connection host traffic features

Att. No.	Attribute Name	Description	Sample data
32	Dst host count	No. of connections (same destination)	150
33	Dst host srv count	No. of connections (same port)	25
34	Dst host same srv rate	Percentage of connections (same services) in 32	0.17
35	Dst host diff srv rate	Percentage of connections (different services) in 32	0.03
36	Dst host same src port rate	Percentage of connections (same source) in 33	0.17
37	Dst host srv diff host rate	Percentage of connections (different destination) in 33	0
38	Dst host serror rate	Percentage of connections (active flag 4) in 32	0
39	Dst host srv serror rate	Percentage of connections (active flag 4) in 33	0

Continued on next page

Table 4 continued

40	Dst host rerror rate	Percentage of connections (active flag 4 REJ) in 32	0.05
41	Dst host srv rerror rate	Percentage of connections (active flag 4 REJ) in 33	0

4. Feature Selection

The process of getting and accessing small and suitable data from collection of large data is called features selection which has a unique interest inside the data mining field. It's significant part to get more performance and accurate results for selected data. Feature choosing, as a process of selecting related depended features for novel features based on some defined conditions, which may be considered as a decrease technique for data mining. Feature selection process contains four primary operations: subset generation, subset estimation, stopping criteria, and final results testing and verification.
The subset generation services is a process act as search method that defined the strategy of the search based on specific candidate feature subsets. The candidate partition is evaluated and compared with the earlier one which removed if the new one is better based on evaluation criteria, this process is repeated until a specific stooping factor completed. The feature selection method can be used in many artificial area with three main categories (Witten, Frank & Hall, 2005) the filter, wrapper, and hybrid models. Features selection has many benefits:
Overfitting reduction: Less redundant data means less opportunity to make decisions based on noise.
Accuracy is increased: Less misleading data means modeling accuracy improves.
Training Time reduction: Less data means that algorithms train faster.

4.1. Genetic Algorithms

GA is a general random search method, able to discovering large search spaces efficiently, which used basically for attribute selection. Genetic algorithms (GA) search from a population of points to find a global maximum or minimum of a function. Unlike many other local search techniques, there is always a chance of escaping from local maxima or minima.
The GAs algorithm is a simulated of the natural systems for evolutions as the principle of "survival of the fittest" given by Charles Darwin (Goldberg, 1989). The processes of the genetic algorithm are:
reproduction: selection process for subset of the input attributes (better string).
Crossover: combination process for the selected attributes for generating better offspring's.
Mutation: modification process for the local string attribute for creating a better string.
The string attribute have two binary bit values, 1 represent selected and 0 represent non-selected, while the population is evaluated and tested for termination of the algorithm at each generation. When the criteria of the termination are not accomplished, the population is run based on the three GA processes and then checked. This process is repeated for specified number of generation.

4.2. Weka Tool

Weka (Waikato Environment for Knowledge Analysis) is a freeware tool designed for machine learning and preprocessing tools such as classification, regression, clustering, association rules, and visualization (Dhanjibhai Serasiya & Chaudhary, 2012).

4.3. Feature Selection Algorithm

The feature selection algorithm contains many steps to select the best related features based on GA algorithm as shown in Figure. 1.

4.4. Selected Attributes

The selected attributes are (15): service, flag, src_bytes, dst_bytes, logged_in, num_root, num_shells, serror_rate, srv_serror_rate, same_srv_rate, diff_srv_rate, srv_diff_host_rate, dst_host_same_src_port_rate, dst_host_srv_diff_host_rate, dst_host_serror_rate.

5. Classification Phase

Neural networks have been ended up being fruitful in mapping input patterns to specific yields. Finding a legitimate arrangement of weights and topology for a system to deliver a craved element can be viewed as an optimization problem.

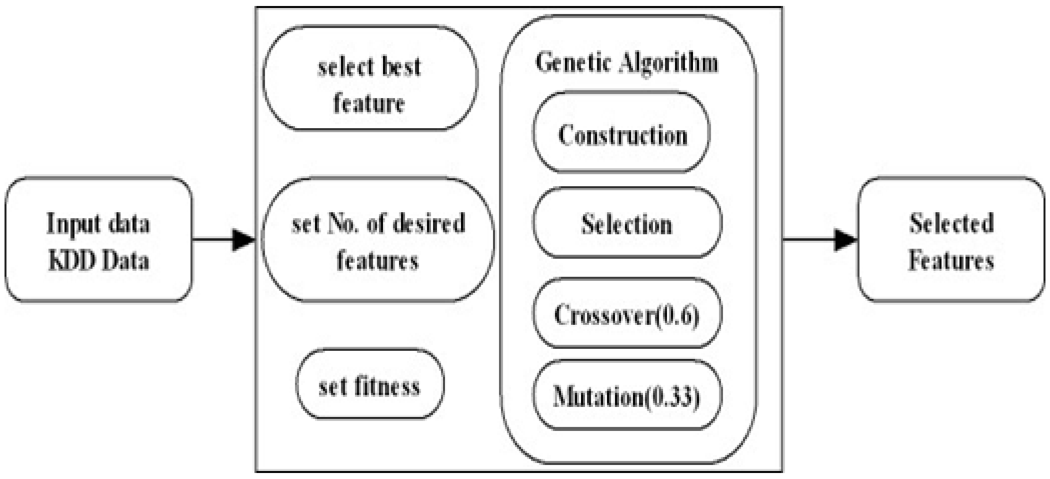

Figure 1. Feature selection steps.

Evolutionary algorithms (EAs) have been appeared to be great at optimization problem solving (Rahmani, 2008). The artificial neuron is charchterized by inputs, having some weights associated with them; an input function which calculates the total net input signal to a neuron coming from all its inputs; and an activation function, which determines the activation level of a neuron as a function of its input. An output signal equal to the activation value is produced through the output of the neuron.
The evolutionary algorithms (EAs) can be used in optimization problem solving using the artificial neural network (ANNs) with ideal design through changing the weights, structure, and learning parameters (Yu, Wang, & Xi, 2008). One of the recent techniques is gravitational search algorithm (GSA). This technique is considerd as heuristic optimization procedure inspired from the law of gravity and mass interactions. It should be remarked that this algorithm has an effiecient capability to search for the global optimum, but expierinces slow convergence in the last iterations.
GSA contains a collection of agents (candidate solutions) that have masses related with objective function value (fitness). All masses attract each other during the generations by the gravity forces between them, the weightier mass has the bigger attraction force. So, the heaviest mass attracted the other masses with the amount of space between this heaviest mass and the attracted masses. GSA agents consists of four parameters: position, inertial mass, active gravitational mass and passive gravitational mass (Mirjalili, Hashim, & Sardroudi, 2012) The position of the mass represents the solution of the problem and the inertial and the gravitational masses are determined by the objective function. Masses are attracted by the heaviest mass. Hence, the heaviest mass presents an optimum

solution in the search space.

Particle swarm optimization (PSO) is another global optimization procedure by Kennedy and Eberhart (1995). PSO inspired by the social behavior of bird flocking. PSO search process used a swarm that contains a number of particles which move to best solution at the search space based on their own best position (Xbest) and the best position of the swarm (Gbest) (Ibrahim, Badr, & Shaheen, 2012). Each particle has two important parameters, velocity (Vi), and its position (Pi) as a possible solution (Mirjalili, Hashim, & Sardroudi, 2012). Each particle starts at a random position in the N dimensional search space of the optimization problem. Determination of the best particle relies upon the objective function which is modeled mathematically by the user. Depending on the velocity of the particle, the (Xbest) and (Gbest) will be updated toward the best solution(Wahba, El Salamouny, & El Taweel, 2015). The velocity (Vi) of each particle is updated as shown in (1). While maximum number of iterations or minimum error criteria is not obtained.

$$v_{in}\,(t+1) = W \times V_{in}\,(t) + C_{1,i}\,r_1\,(X_{ibest}\,(t) - P_i\,(t)) + C_{2,i}\,r_2\,(X_{gbest}\,(t) - P_i\,(t)) \tag{1}$$

Where:

Vin (t+1): The new velocity of particle i in the dimension n.

W: The weighting function (considered as a random number within [0,1]).

Vin: Old velocity of particle i in the dimension n.

Cj,i: Acceleration coefficient for the particle i.

C1,I: Cognitive learning value for particle i.

C2,i: Social learning value for particle i.

r1,r2: Random numbers within [0,1].

Xibest: Optimum or best position that particle iin dimension n at time t.

Xgbest(t): Optimum or best solution found by the entire population at time t.

Pi (t): Current position of the ith particle at generation t.

The position of the particle is updated using Equation (2) as follows

$$X_i\,(t+1) = X_i\,(t) + V_i\,(t) \tag{2}$$

Where:

Xi (t+1) New position of particle i in the dimension n.

Xi (t) Old or previous position for particle i at time t.

Vi (t) Denotes the old velocity of particle i in the dimension n.

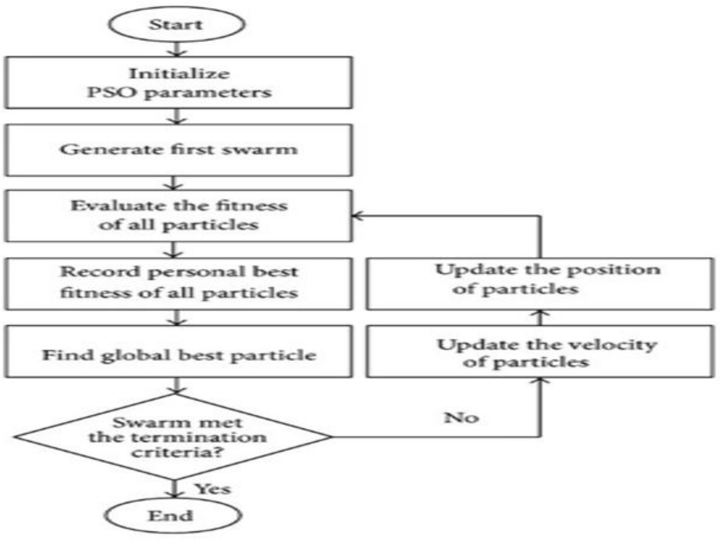
Start
Initialize PSO parameters
Generate first swarm
Evaluate the fitness of all particles
Record personal best fitness of all particles
Find global best particle
Swarm met the termination criteria?
Yes
End
No
Update the velocity of particles
Update the position of particles

Figure 2. Steps of the PSO Algorithm

PSO starts in a random way by placing the particles in a search space related to a specific problem. At each iteration step, the velocities of particles are calculated using Equation (1). The positions of particles are calculated by Equation (2) after the velocities are defined. This process is repeated until stopping criteria is accomplished. The main steps of PSO algorithm are indicated in Figure 2 above.

Gravitational Search Algorithm is a new experiential optimization method which inspired from Newton's theory that states: "Every particle in the universe attracts every other particle with a force that is directly proportional to the product of their masses and inversely proportional to the square of the distance between them" (Newton, 1729).

By generations, all the GSA masses which related to the objective function value are attracted to each other through the gravity forces, and the heavier mass close to the global optimum attract.

The GSA algorithm parameters begin at random by setting all the agents in the search space. Throughout iterative procedure, the gravitational forces from agent *j* on agent *i* at a specific time *t* are updated by Equation (3) as follows:

$$F_{ij}^{d}\ (t) = G(t)\frac{M_{pi}\ (t) \times M_{aj}\ (t)}{R_{ij}\ (t) + \varepsilon}(X_j^d\ (t) - X_i^d(t)) \tag{3}$$

Where:

M_{aj} is the active gravitational mass related to agent j,

M_{pi} is the passive gravitational mass related to agent i,

G(t) is gravitational constant at time t,

ε is a small constant, and

$R_{ij}(t)$ is the Euclidian distance between two agents i and j.

The G(t) is computed by Equation (4):

$$G(t) = G_0 \times exp(-\propto\ \times \frac{iter}{maxiter}) \tag{4}$$

Where $\propto$ and G_0 are descending coefficient and initial value respectively, and

iter : is the current iteration,

maxiter: is maximum number of iterations.

In a search space with the dimension *d*, the total force that acts on an agent *i* is computed using (5):

$$F_i^d(t) = \sum_{j=1, j\neq 1}^{N} rand_j\ F_{ij}^d(t) \tag{5}$$

Where $rand_j$ is a random number in the interval [0,1].

Baesd on the law of motion, the acceleration of an agent is proportional to the result force and inverse of its mass, therefore the acceleration of all agents can be calculated using Equation (6):

$$ac_i^d(t)\ =\ \frac{F_i^d(t)}{M_{ii}(t)'} \tag{6}$$

Where t is a specific time and Mi is the mass of object i.

The velocity and position of agents are calculated as follows Equation (7) & (8):

$$Vel_i^d(t+1) = rand_i \times Vel_i^d(t) + ac_i^d(t) \tag{7}$$

$$x_i^d(t+1) = x_i^d(t) + acVel_i^d(t+1) \tag{8}$$

Where $rand_i$ is a random number located in the interval [0,1].

All masses in GSA are initialized with basic values, then the velocities values for all masses are determined by (7), while the gravitational constant, total forces, and accelerations values are obtained using relations (4) and (5) independently one by one at the same time. The positions of masses are determined by (8). Lastly , the GSA will not stop until the end criteria are accomplished. The steps of the GSA algorithm are described in Figure 3.

A hybrid PSO with GSA are used using low-level co-evolutionary heterogeneous method's. The basic idea of PSOGSA is to combine the ability of social thinking (gbest) in PSO with the local search capability of GSA using Equation (9).

$$v_i\,(t+1) = W \times V_i\,(t) + C'_1 \times rand \times ac_i(t) + C'_2 \times rand(gbest - x_i\,(t)) \tag{9}$$

Where $V_i(t)$:is the velocity of agent i at iteration t,

C_j' :is a weighting factor,

w : is a weighting function,

$rand$: is a random number between 0 and 1,

ac$_i$(t): is the acceleration of agent I at iteration t, and

$Gbest$: is the best solution so far.

The positions of particles are updated on each iteration, using Equation (10):

$$X_i\,(t+1) = x_i\,(t) + v_i\,(t+1) \tag{10}$$

All agents in PSOGSA are randomly initialized. Then the Gravitational force, gravitational constant, and resultant forces, among agents are determined by Equation (3), (4), and (5), and the accelerations of particles are determined by Equation (6). At each iteration, the best solution only will be updated and the velocities values of all agents are determined by (9). At last, the positions of all agents are calculated using Equation (10).

The value of solutions (fitness) in PSOGSA is updating and the agents near by appropriate solutions attempt to induce the other agents in the search problem. The agents move very slowly when they are near the good solution. PSOGSA use a memory (gBest) to save the best solution has found so far. By correcting the C_1' and C_2' values, the capabilities of global search and local search can be equally adjusted , and each agent can detect the best solution and go near it.

6. The proposed model

In this paper, both particle swarm optimization (PSO) and gravitational search algorithm (GSA) wereadapted to neural network to get the optimum weights and biases that give minimum error and higher classification rate for the feed-forward neural network (FFNN). With the aim of investigating the algorithm FFNN-PSO or FFNN-GSA, the following essential parameters have to be clarified. First, an objective function using the error of the FFNN should be introduced to determine particles' fitness in FFNN-PSO and agents' in FFNN-GSA. Second, an encoding strategy should be defined to encode the weights and biases of the FFNN for the particles of the FFNN-PSO and the agents of the FFNNGSA. All steps of the proposed model are shown in Figure 4.

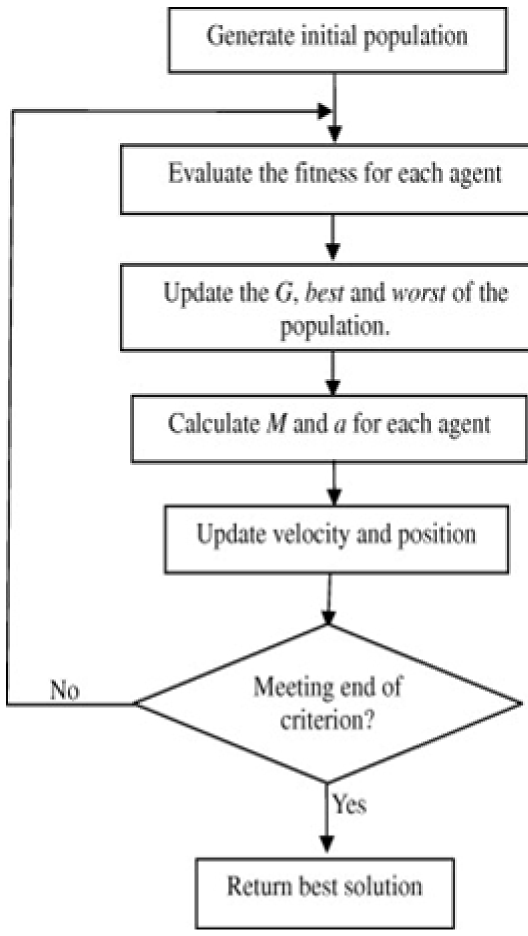
Generate initial population
Evaluate the fitness for each agent
Update the *G*, *best* and *worst* of the population.
Calculate *M* and *a* for each agent
Update velocity and position
Meeting end of criterion?
No
Yes
Return best solution

Figure 3. Steps of the GSA Algorithm

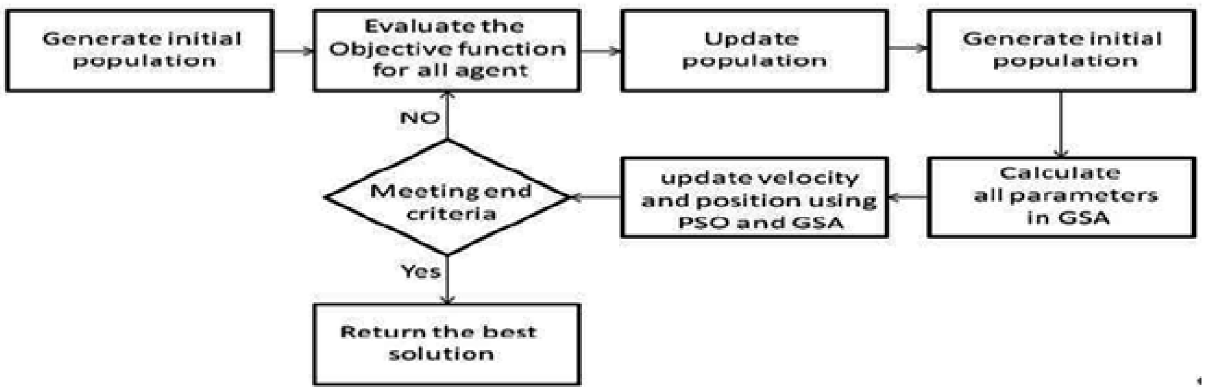
Generate initial population
Evaluate the Objective function for all agent
Update population
Generate initial population
Calculate all parameters in GSA
update velocity and position using PSO and GSA
Meeting end criteria
NO
Yes
Return the best solution

Figure 4. Steps of the Hybrid PSOGSA Algorithm

6.1. The related objective function

The objective function can be described as follows:

The neural network FFNN has three layers (1 input with n nodes , 1 hidden with h nodes, and 1 output layer with h nodes). The output of each hidden node at each epoch of learning process is given by Equation (11) , (12):

$$f(s_j) = 1/(+exp(-(\textstyle\sum_{i=1}^{n} w_{ij}x_i + b_j))) \tag{11}$$

for j=1,2,. . . .,h

$$s_j = \textstyle\sum_{i=1}^{n} w_{ij}x_i + b_j \tag{12}$$

Where (n) value is the number of the input nodes,(wij)value is the connection weight from the ith node in the input layer to the jth node in the hidden layer, bj is the bias (threshold) of the jth hidden node, and xi is the ith input. After calculating outputs of the hidden nodes, the final output can be defined in Equation (13):

$$o_k = \textstyle\sum_{i=1}^{k} w_{kj} f(s_j) + b_k \tag{13}$$

Where (wkj) value is the connection of weight from the (jth) hidden node to the (kth) output node, and (bk) is the bias (threshold) value of the (kth) output node. The learning error E_k is obtained by Equation (14)

$$E_k = \textstyle\sum_{i=1}^{m} (o_i^k - d_i^k)^2 \tag{14}$$

Where (q) value is the number of training samples, (dik) value is the desired output of the ith input item when the (kth)vlaue is used, and (yik) is the real output value of the ith input item when the(kth)value is used.So, the fitness function of the (ith) training example using Equation (15):

$$Fit(X_i\ ) = E(X_i\ ) \tag{15}$$

6.2. Equations of GSA

6.2.1. Setting the Agents

Setting the position value of (N) number of agents randomly by (16):

$$X_i = (x_i^l, ..., x_i^d, ..., x_i^n) \tag{16}$$

for i=1,2,....,N

X$_i$ represents value the positions of the i$_{th}$ agent in the d$_{th}$ dimension, while n is the space dimension.

Calculating the best and worst fitness for all agents at each iteration for minimization problems so that to achieve the fitness evolution value by Equation (17), (18):

$$best(t) = min\ f\ it_j\ (t) \tag{17}$$

$$worst(t) = max\ f\ it_j\ (t) \tag{18}$$

j=1,....,N

The $F_{it_j}(t)$ value is the fitness value of the (jth) agent at iteration t, the best (t) value is the best fitnessat iteration t, while the worst (t) value is the worst fitnessat iteration t.

6.2.2. Estimating the gravitational constant

The gravitational constant value $G_c(t)$ at iteration t is given by Equation (19):

$$G_c(t) = G_o\ e^{\alpha(-t/T)} \tag{19}$$

Where : (α) value is the descending coefficient, (G$_o$)value is the initial gravitational constant, (*t*) value is the current iteration, while (*T*) value is the maximum number of iterations.

6.2.3. The masses of agents are calculated as follows:

The gravitational and inertia masses values for each agent are at iteration (*t*) are obtained using Equation (20) , (21) and (22):

$$M_{ai}\ =\ M_{pi}\ =\ M_{ii}\ =\ M_i\ for\ i = 1,2,..N \tag{20}$$

$$m_i(t) = \frac{f\ it_i(t) - worst(t)}{best(t) - worst(t)} \tag{21}$$

$$M_i(t) = \frac{m_i(t)}{\sum_{j=1}^{N} m_j(t)} \tag{22}$$

The active and passive gravitational masses values are M$_{ai}$ and M$_{pi}$, and the inertia mass value of the i$_{th}$ agent (Miiis), and the mass value of object i is M$_i$. Acceleration of agents is obtained by determining the values of $F_i^d(t)$ (the total force acting) on i$_{th}$ agent by Equation (23) , (24):

$$F_i^d(t) = rand_j\ F_{ij}^d(t) \tag{23}$$

$$F_{ij}^d = G_c(t).(M_{pi}(t) \times M_{aj}(t)/R_{ij}(t) + \varepsilon).(x^d{}_j(t) - x_i^d(t)) \tag{24}$$

Where ε is a small constant and Rij is the Euclidian distance between two agents *i* and *j* and it is calculated using Equation (25):

$$R_{ij}(t) = \|X_i\ (t), X_j\ (t)\|_2 \tag{25}$$

Acceleration of the ithagents a$_i$ at iteration t is obtained from Equation (26):

$$a_i^d(t) = F_i^d(t)/M_{ii}(t) \tag{26}$$

6.2.4. Updating the speed and positions of agents:

Determining the speed and the position of the agents at the next iteration *(t+1)* is achieved by Equation (27) , (28) as follows:

$$v_i^d(t+1) = rand_i \times v_i^d(t) + a_i^d(t) \tag{27}$$

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1) \tag{28}$$

Social thinking capability (gbest) of PSO algoirhtm and local search capability of GSA were grouped together in PSOGSA. All agents are initialized randomly, then the values of gravitational force, gravitational constant, and resultant forces are calculated . The accelerations of elements are defined in Equation (20).At each iteration, the best solution is updated and the velocities values of all agents are calculated using Equation (27). At last, the agents positions are updated using Equation (28). Updating the value of velocities and the positions will stop when conditions are met.

7. Simulated results and model verification

Exermienatl reuslts was run on Matlab 2015a environment using core *i5* processor, and 4 GB of Ram.The last stage in the intrusion detection system is classification. In this work the classification stage is implemented using FFNN-GSA and FFNN-PSO.

Table 5 show the training results of the feed-forward neural network with the opmtiziation techqniue GSA and PSO for the intrusion detection process based on 16626 record of training NSL-KDD dataset samples. The reuslts are measured by the mean square error (MSE) and the classification rate. With many number of iterations the neural network structure of FFNN for both PSO and GSA are as follows: 2-layers input neural network with 71 neurons, hidden layer with 8 neurons, and output layer with 10 neurons. At the start of trainig neural network the weights are initialized randomly, which may casue corrupt recognition that improved after number of iteration. By increasing the number of iterations, the MSE decreases, the classification rate increases and the time consumed increases. Increasing the swarm size and the number of masses enhances the classification rate.

Table 5 illustrates the test accuracy of the classification model with and without the GA feature selection technique. The combination of GSA-PSO feed forward algorithm has the best recognition rate against other algorithms with GA feature selection, while FNN PSO algorithm has the best result without the GA feature selection. It should be noted that, dealing with 15 features is better than dealing with 41 features in classification, MSE error, and processing time.

Table 5. Comparison of Recognition Rate & MSE values of various classification algorithms

Classification Algorithm (2000iterations)	Test Accuracy (%) with 41 Features			Test Accuracy (%) with 15 Features		
	Recognition Rate (%)	**MSE**	**Time(Seconds)**	**Recognition Rate (%)**	**MSE**	**Time(Seconds)**
FNN PSO	94.0667	0.0060613	2237	96.9333	0.053256	895
FNN GSA	81.675	0.093597	2500	97.2	0.042981	1000
FNN GSA-PSO	90 .5	0.13776	3367	98.966	0.00091032	1347

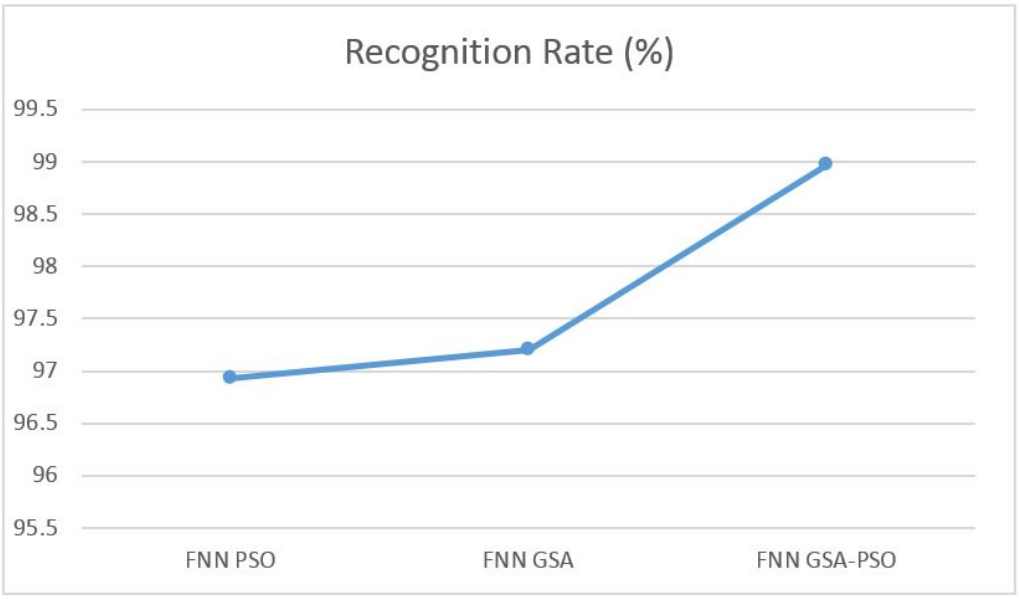

Figure 5. The Recognition Rate for the Classification algorithms

8. Conclusion

The intrusion detection systems are utilized to distinguish the attacks coming towards the basic information of the client. An introduced optimized neural network model is based on studying the intrusion detection features and select the most important features only which creating an optimal intrusion detection system. This system is based on combination of GSA and PSO with feed forward neural network FFNN classifier with feature selection. The classification proposed system is established by three different optimization algorithms with certain GA as feature selection method in WEKA program. Getting the optimum performance for the classification process is our major goal, which has been done. The best algorithm performance based on GA feature selection is FNN GSA-PSO classifier with an actuary of 98.96 % and The process time is 1347 sec. for the model. While FNN GSA-PSO without any feature selection algorithm has the lowest average error 0.13776compared to others. Generating an intelligent agent fire toward the infected packet with other new different classification algorithms in a real time environment will be our future work.

9. References

1. Aghdam, M. H., & Kabiri, P. (2016). Feature Selection for Intrusion Detection System Using Ant ColonyOptimization. International Journal of Network Security, 18(3), 420-432.
2. Ahmad, I., Abdulah, A. B., Alghamdi, A. S., Alnfajan, K., & Hussain, M. (2015). Feature Subset Selection for Network Intrusion Detection Mechanism Using Genetic Eigen Vectors. International Conference on Telecommunication Technology and Applications (CSIT), 5.
3. Brunswick, U. O. (2017). Retrieved from http://nsl.cs.unb.ca/NSL-KDD/
Dhanjibhai Serasiya, S., & Chaudhary, N. (2012). Simulation of Various Classifications Results using WEKA. International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878, 1(3), 155-160.
4. Goldberg, D. (1989). Genetic Algorithms in Search, Optimization, and Machine learning. Addison Wesley.
5. Ibrahim, H. E., Badr, S. M., & Shaheen, M. A. (2012). Adaptive Layered Approach using Machine Learning Techniques with Gain Ratio for Intrusion Detection Systems. International Journal of Computer Applications (0975 – 8887), 56(7), 10-16.
6. Ibrahim, L. (2010). Anomaly network intrusion detection system based on distributed time-delay neural network(DTDNN). Journal of Engineering Science and Technology, 5(4), 457 – 471.
7. Kaushik, S. S., & Deshmukh, P. R. (2011). Detection of attacks in an intrusion detection system. International Journal of Computer Science and Information Technologies (IJCSIT), 2(3), 982-986.
8. Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. Proc. IEEE Int. Jt. Conf. neural networks, 4(6), 1942–1948.
9. Majeed, S. K., Hashem, S. H., & Gbashi, I. K. (2013). Propose HMNIDS Hybrid Multilevel Network Intrusion Detection System. IJCSI International Journal of Computer Science Issues, 10(5), 200-208.
10. Mirjalili, S., Hashim, S. Z. M., & Sardroudi, H. M. (2012). Training feedforward neural networks using hybrid particle swarm optimization and gravitational search algorithm. Appl. Math. Comput., 1125–11137.
11. Newton, I. (1729). In experimental philosophy particular propositions are inferred from the phenomena and afterwards rendered general by induction. Andrew Motte's English translation.
12. Parvat, T. J., & Chandra, P. (2015). Modeling Intrusion Detection Systems with Machine Learning And Selected Attributes. Journal of Multidisciplinary Engineering Science and Technology (JMEST), 2(6), 1553-1557.
13. Rahmani, M. (2008). Particle swarm optimization of artificial neural networks for autonomous robots. Chalmers University of Technolog.
14. Sadek, R. A., Soliman, M. S., & Elsayed, H. S. (2013). Effective Anomaly Intrusion Detection System based on Neural Network with Indicator Variable and Rough set Reduction. IJCSI International Journal of Computer Science Issues, 10(6), 227-233.
15. Sharma, N., & Mukherjee, S. (2012). A Layered Appoach To Enhance Detection Of Novel Attacks In IDS. International Journal of Advances in Engineering & Technology, 4(2), 444-455.

16. Tang, H., & Cao, Z. (2009). Machine Learning-based Intrusion Detection Algorithms. Journal of Computational Information Systems(5:6), 1825-1831.
17. Wahba, Y., El Salamouny, E., & El Taweel, G. (2015). IJCSI International Journal of Computer Science Issues, 12(3), 255-262.
18. Wang, D., Yeung, D. S., & Tsang, E. C. (2007). Weighted Mahalanobis Distance Kernels for Support Vector Machines. IEEE Transactions on Neural Networks, 18(5), 1453-1462.
19. Witten, I. H., Frank, E., & Hall, M. A. (2005). Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann, Elsevier Inc.
20. Yu, J., Wang, S., & Xi, L. (2008). Evolving artificial neural networks using an improved PSO and DPSO. Neurocomputing, 71(4-6), 1054–1060.