Optimization of BPN Parameters Using PSO for Intrusion Detection in Cloud Environment

Sajith P J (pjsajith@gmail.com)
Sathyabama Institute of Science and Technology https://orcid.org/0000-0001-7514-7662
G Nagarajan
Sathyabama Institute of Science and Technology

Research Article

Keywords: Back Propagation Network, Intrusion Detection System, Neural Network, Particle Swarm Optimization, System Calls

Posted Date: February 3rd, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1298053/v1

Version of Record: A version of this preprint was published at Soft Computing on June 21st, 2023. See the published version at https://doi.org/10.1007/s00500-023-08737-1.

Abstract

The usage of internet is getting increased in all aspect, like from building various models that are fully connect with internet to the usage of digital media for 24/7. As this is rising in a way, on the other side the concern about “data security” is raising and everybody’s information needs to be protected from any other attacks or can say there shouldn’t be no data leakage. So, for detecting these attacks, intrusion detection system is placed. But placing this traditional Intrusion Detection System (IDS) will increase the concerns of security even more, so to amp the process integration of latest technology such deep learning comes in action. Thereby this paper proposes an IDS using Back-propagation Network (BPN) where intrusions are identified based on the system calls that we collected in the dataset KDD cup 99. Also, to increase the optimization of neural network, we used Particle Swarm Optimization (PSO) thereby increase the accurate detection of the system saying if that is normal or abnormal in behavior. We evaluate our proposed model with other methods like ANFIS, F-GNP, FCM in which the proposed model gives 96.5% accurate detection.

1. Introduction

The issue of ensuring data has existed since data has been overseen. Be that as it may, as innovation advances and data the board frameworks become increasingly incredible, the issue of authorizing data security additionally turns out to be more basic [1]. The development of this electronic climate accompanies a comparing development of electronic wrongdoing where the system is utilized either as a device to carry out the wrongdoing or as an objective of the wrongdoing [1].Many networks did not consider insurance to guarantee against network attacks. That is why many networks have been hacked in recent times. The inability to achieve their framework puts many organizations and associations at serious risk of misfortune. Generally, a solitary assault can cost a large number of dollars in expected income. Also, that is only the start. The harms of assaults incorporate not just loss of licensed innovation and risk for bargained client information (the time/cash spent to recuperate from the assault) yet in addition client certainty and market advantage. The security of systems and organizations needs to be improved to protect the infrastructure from hazards [2–5]. Joining the escalation of electronic errors, the Secure Data Foundation's plan, for example, is gradually becoming a test of the Interruption Recognition Framework (IDS) for preventing and identifying events.. Figure 1 delineates the interruption identification framework and outer/inward organization interruption assaults. Outfitted for recognizing awful interruptions is the prescient model (for eg: a classifier), to assemble it is the interruption identifier learning task what's more, typical associations. As of late, extensive research has been done to apply neural organizations to identify barriers. An ANN contains a collection of components which are deeply interconnected. Provide a collection of information sources and the desired yield, and the transition from donation to yield is governed by the interrelated loads in the component preparation. By adjusting these interrelationships, the organization can match the appropriate revenue. In IDS the ability of the high capacity to carry learning for visual demonstration makes neural organizations adaptable and stunning.

Although, the time taken to run the model from a big dataset is very large. The aggressive behavior of the IDS can be accurately expected for this.

There are two kinds of interruption discovery frameworks are accessible. Host based Intrusion Detection System (HIDS) which utilizes data accumulated from a solitary host, for example, review trail, log records, framework call groupings and so forth NIDS utilizes the assembled information by breaking down the system organization traffic. HIDS regularly screens and investigations the single host occasions, for the most part the host-based models utilize rule-based example coordinating with approaches [9][10]. For every client a profile is made and the HIDS persistently screens and thinks about the current review record and the current client profiles. On the off chance that there is deviation over specific limit esteem then the current movement is considered as a vindictive action. The expanded utilization of the systems and system related assets, brought about expanded number of gadgets and clients associated with the organizations. A Network based Intrusion Detection System (NIDS) screens and examines the organization traffic to shield a framework from network-based malignant exercises. NIDS works at chosen framework on an organization and which examinations network traffic, bundle by parcel to recognize interruption. On the off chance that NIDS is introduced in the organization, it diminishes the responsibility of interruption identification on each individual framework (Figure 2).

1.1 Key Highlights

This paper focus over building a Intrusion detection system which is been integrated with DL could potentially analyze behavior of attacks in which following are some key notes;

a. Intrusion detection system using BPN
b. With collection of system calls that is been passed over this network for analyzing the behavior
c. For improving the optimization power of neural networks, PSO is been utilized
d. The proposed system is compared with other models like ANFIS, F-GNP, and FCM in which the proposed model outperformed with accuracy 96.5%.

Organization of paper: As we already come across the overview and the types of IDS in Section 1, rest is as follows; Section 2 depict related works based on IDS integrated with DL then followed by Section 3 with methodology, Section 4 with Implementation and Result, and at last section 5 with conclusion.

2. Related Works

In [11] the creator proposed an information mining structure for building interruption identification models. The main thought is to process the programs of information mining specifically, grouping, meta-learning, affiliation manages, and continuous scenes to review information for figuring abuse and irregularity location models that precisely catch the behavior (i.e., designs) of interruptions and ordinary exercises. In spite of the fact that, proposed discovery structure can identify a greater level of new and old U2R and PROBING assaults, it missed countless new DOS and R2L assaults. In [12], it is generally centered around information mining procedures that are being utilized for the purposes of such, and

afterward introduced a groundbreaking thought on how information mining can help IDSs by using bi-clustering as an apparatus to investigate the traffic of network and upgrade IDS's. An investigation of scholastic exploration utilized the accepted standard benchmark information, to improvise the efficiency of interruption identification rate of KDDCup 99. In third International Knowledge Discovery and Data Mining Tools competition, KDDcup 99 was used. The information was generated in such a way as to handle the 1998 DARPA Interruption Location (ID) Assessment Organization's tcpdump information. The challenge was, to make a prescient structure, to arrange the organization associations into 2 classes: Attack or Normal. Assaults are ordered into the ('DoS') Denial of Service, ('U2R') User-to-Root, 'Test', ('R2L') Remote-to-Local, classes. The digging review information for the models of mechanized for (MADAMID) ID was utilized as highlight development structure in KDDCup 99 rivalry [13]. MADAMID yields 41 highlights: initial 9 highlights are fundamental highlights of a bundle, content highlights are 10-22, highlights of traffic are 23-31, and based highlights are 32-41 has. Decisions of the accessible dataset are: full dataset and 10% corresponding information. The nitty gritty assessment after effects of KDDCup 98 and KDDCup 99 test was distributed in [14]. The Third International Knowledge Discovery and Data Mining Tools Competition task continued as the basic work. Machine learning solutions could be found from it. Most of the published works used only 10% of the training and testing data. Very few custom built dataset was used. With the dataset [17] of KDDcup 99 a recent survey was conducted on the id of machine learning based. Most of the results published on KDDcup 99 have been utilized for dimensionality reduction in featured engineering methods [16]. Most of the newly available machine learning used the same dataset. Few studies have used the custom-built dataset. The outputs are partially comparable to the KDDcup99 contest. In [18], by the use of Naive Bayesian network explored the Bayesian networks for ID, in which leaf node = features and root node = class connection. Later, [20] to ID, the application of the Naive Bayes network is identified and by means of detailed experimental analysis, challenge of KDDCup 99 the winning entries with which compared, in 'Probe' and 'U2R' categories better performance is given by a Bayesian network. In [21], on parison-window estimators based method of non-parametric density estimation was studied by the use of Normal distribution and Gaussian kernels. Model temporal and spatial data were used to identify complex anomalies. NIDS proposed a genetic algorithm for this[22]. Using System Particle Optimization, Agent Colony Optimization and Colony Clustering [19] for ID, a set of intelligence techniques and techniques of synchronization learning overview is provided.

3. Methodology

Figure 3 depict the overall workflow of proposed model in which the system call is collected from the KDD cup 99 dataset are given for the pre-processing stage. Here all the system calls are collected in raw fashion, by using the sliding window mechanism these are pre-processed and are passed for feature selection in which for better optimization we use Particle Swarm Optimization and then passed over to the decision network were using the selected features, neural network (BPN) will process these features into several layer and give the result as system behaves properly or not.

3.1 Dataset Description

For this model for execution, we utilized KDD cup 99 dataset created in MIT Lincoln Laboratories. The Dataset is made by presenting physically produced network-based assaults. Different attacks that can be potentially found in an organization is characterized in a brief form concerning KDD interruption discovery evaluation dataset[6]. System will be analyzed at different levels such as system accuracy, system ability and cost to differentiate abnormal behavior and normal behavior. IDS can work based on either privileged process behavior or user behavior. The privileged processes have the privileges to access and use system resources. All the normal system calls are gathered in the normal trace step. In abnormal trace step, abnormal system call sequences are gathered. Data set of KDD cup 99 is utilized for collecting abnormal and normal traces. The stide, xlock, ps and login processes sequences of system calls are collected from KDD cup 99 data set. The repeated execution of these processes generates system call sequences which are recorded in separate files. Each trace system call sequence contains ten to thousand system calls. These traces are collected while there are no malicious activities. The examples of abnormal processes are iprcp, buffer overflow, sun sendmailcp etc.

Another example is Syslog attack. It uses the interfaces like syslog which makes buffer overflow in send mail. Intrusion traces contains three sunsendmailcp attacks, forwarding loops five error conditions, two traces of syslog-local attacks, the syslog-remote attacks of two traces, and an attacks of decode two traces. Each trace contains two attributes: process ID and a system call value. The process ID is used to identify the specific system call. An abnormal process will not have the sequences of normal system calls (Figure 4). The current sequence of system calls can be compared with the sequence of normal system calls stored and deviations can be detected[7][8].

3.2 Data Pre-processing

After collecting the system call sequences of from the active process, the next step is preprocessing of data. The gathered information about system call is basic raw collection data. The techniques used for preprocessing have to be applied on raw data to make the data set into processing dataset. A unique number will be assigned to each and every system call name. For instance, 8 for open, 9 for close, 74 for mmap etc. The unique numbering will make it is easy to access the system call, reduces data complexity and convenient format for processing. With proper sliding window mechanism, long system call sequence numbers can be processed. The normal behavioral data base uses the window size of 3. For example, the normal behavior database can be created from the following system call sequence Open, read, mmap, mmap, open, read, close for the given sequence, the system calls will be put in position 1, position 2 and position 3 as shown in below table. The window size decides the pairs generated. Table 1 depict the sequence of system call in proposed system.

Table 1
System call sequence

Current	Position1	Position2	Position3
$Open_1$	$Read_1$	$mmap_1$	$mmap_1$
$Read_1$	$mmap_1$	$mmap_1$	
$mmap_1$	$mmap_1$		
$Read_1$	$mmap_1$	$mmap_1$	$Open_1$
$mmap_1$	$mmap_1$	$Open_1$	
$mmap_1$	$Open_1$		
$mmap_1$	$mmap_1$	$Open_1$	$Read_1$
$mmap_1$	$Open_1$	$Read_1$	
$mmap_1$	$Open_1$	$Read_1$	$Close_1$
$Open_1$	$Read_1$	$Close_1$	
$Read_1$	$Close_1$		

By analyzing the data set it is found that certain system calls are executed frequently. These systems call executions may be followed by different system calls. For example, the read is followed by different system calls and executed two times. Therefore, all system calls are recorded first and then, expanded the database for different sequences. The expanded format is given in the following Table 2.

Table 2
Expanded system call

Current	Position1	Position2	Position3
$Open_1$	$Read_1$	$mmap_1$, $close_1$	$mmap_1$
$Read_1$	$mmap_1$, $close_1$	$Mmap_1$	$Open_1$
$mmap_1$	$mmap_1$, $open_1$	$Open_1$, $read_1$	$Close_1$, $read_1$

Using the sliding window, many system call sequences are produced and stored in database. After data base is preprocessed from raw information, normal behavior rule can be easily formed from this data set.

3.3 Particle Swarm Optimization

Here once it is normalized, these are now passed over to feature selection process where you naturally or physically select those highlights which contribute most to your expectation variable or yield in which you are keen on. So, for that PSO is used here for feature extraction. PSO is an equal estimation, which has

the advantages of straightforward execution, high exactness and quick assembly[19]. To track down the best arrangement, PSO instates some irregular arrangements in arrangement space, these arrangements are a few particles, where characterize the molecule speed v_i and the molecule position x_i. In the meantime, use the capacity of health to determine if the circumstances of the particles are ideal, use *pbest* and *gbest* to capture the individual best circumstances and social opportunity independently. For every particle, note its well-being, it will also be *pbest* if it is better contrasted with *pbest*, and it will be like *gbest* expect better contrasted with *gbest*, update the speed and location of the molecule. The speed of molecules and position update rules are according to the accompaniment;

$$v_i = wv_i + c_1 \times rand\,() \times (pbest_i - x_i) + c_2 \times rand\,() \times (gbest_i - x_i) x_i = x_i + v_i$$

1

where v_i is the speed of the molecule, *w* is inactivity weight, *rand()* is an irregular worth somewhere in the range of 0 and c_1 and c_2 is the current situation of the molecule c_1 and c_2 are speed increase factor. If the velocity or circumstance of the particles exceeds the degree of stroke, it will be defined as the most limiting velocity or the circumstance of the cutoff. At the point where the molecule has been reinvigorated, it will keep reheating until the best game plan is found. Regularly finding the best position or appearing at the most remarkable number of cycles will halt the demand. In BPN, the number of concealed core layer points affects the generation of the independent learning stage and the fine-tuning of coordinated learning stage. Along these lines, the quantity of covered up layer hubs in the profound adapting should be enhanced by PSO calculation to improve the exhibition of the organization.

3.4 Back Propagation Network

Learn an example for back propagation network. When you provide examples of networking algorithm and it changes the weight of the network, for a particular input the required output will given when completed the training. For simple pattern identification and mapping tasks Back Propagation Networks can be used. As mentioned now, you need to give a particular input in order to get the desired output. It is mentioned in Figure 5.

If it is the first pattern to the network, we would like the output to be 0 1 as shown in Figure 6. (yellow line =1 and black= 0 like previous examples). Training Pair means the input and its corresponding target.

Once the network is trained, it will provide the desired output for any of the input patterns.

If the network is trained once then it will yield the required output to any input. Now let's see how it goes. All weights must first initialize the network by giving small random numbers (between -1 and 1). Now you need to perform the forward pass (give the input and calculate the output). The calculations will give you a different output than you need (the target), all weights will be random. Then each neuron’s error is calculated. Here, Actual Output = Target (i.e. what you actually get - what you want). The error get from the output is then used for changing the weight. Then we can reduce the error part. By this way each

neuron’s output will get nearer to required output value. It is known as reverse pass. This step is iterated until the error is minimal.

Algorithm1: BPN

1. Give the information and take the yield from the organization. Since the main weight is irregular numbers, hence recollect that the principal yield can be anything.

2. Presently we need to address the blunder of neuron B. Blunder is the thing that you need – What you really get, all in all: ErrorB = OutputB (1-OutputB) (TargetB – OutputB) The "Yield (1-Output)" term is fundamental in the condition due to the Sigmoid Function – in the event that we were just utilizing a limit neuron it would simply be (Target - Output).

3. Presently you need to alter the weight. Let WAB = introductory weight and W+AB = trained (new) weight. W+AB = WAB + (ErrorB x OutputA). Notice that it is the yield of the interfacing (neuron A) we use (not B). This is the way we update every one of the heaps on the yield layer.

4. Figure the Errors for the secret layer neurons We can't ascertain it straightforwardly from the yield layer since we don't have an objective. That is the reason this calculation is known by that name). This is finished by taking mistakes from the need yield neurons and by means of the heap, running them back to get errors of covered up layer. For model assuming neuron An is associated as to B and C as appeared, for creating a blunder for A we need take the blunders from B and C. ErrorA = Output A (1 - Output A) (ErrorC WAC + ErrorB WAB) Again, the factor "Yield (1 - Output)" is available due to the sigmoid crushing capacity.

5. Assuming you get the mistake for the hidden layer neurons once, the subsequent stage to change the secret layer weight. Consequently we can rehash this strategy and make it workable for quite a few layered networks. Once in a while there might be questions about its capacity. It shows the estimation of a FCN. It comprises, number of data sources =2, covered up layer neurons = 3 and yield = 2. Where w + = new and recalculated weight, w (without addendum) =old weight. The opposite interaction can be determined similarly.

4. Implementation And Results

Initially this model is implemented over wamp server and are run in python environment over the i5 Core intel system. The proposed model is evaluated under two performance measure such as Detection Rate (DR) and False Alarm Rate (FAR), also it is compared with other models such as ANFIS, F-GNP and FCM.

a. Detection Rate: Detection rate indicates, among all attack data, the percentage of detected attack, and is provided as,

DR: TP/ (TP + TN) *100

b. False Alarm Rate: False Alarm Rate is otherwise known as false positive(FP). It is the proportion that normal data is falsely detected as attack behavior. Accuracy is classified as: true positive(TP) and true negative(TN).It is the proportion of correctly classified data.

Table 3
Performance Measure

Models	FAR	DR (%)
ANFIS	3.4	92
F-GNP	1.9	80
FCM	2.4	85
BPN	4.4	96.5%

Table 3 shows the comparison of various systems like ANFIS, F-GNP and FCM with our system under the performance measure DR and FAR. The FAR of our system is 4.4 while it is 1.9 for F-GNP. While taking DR our system shows a detection rate of 96.5% which is better than all the other compared models.

ANFIS: The ANFIS structure maps contributions through input enrollment works and related boundaries, and afterward through yield participations and related boundaries to yields. During the learning interaction, the boundaries related with enrollment capacities changes. An incline vector empowers the figuring of these limits, giving an extent of how well the FIS models the data/yield data for a given game plan of limits. In the wake of procuring the tendency vector, any of the couple of smoothing out timetables could be applied to change the limits for reducing some mix-up measure. This learning method works correspondingly as that of neural associations. When appeared differently in relation to the generally FIS, ANFIS is really astounding. This makes the fuzzy system to acquire from the data they model (Figure 7).

F-GNP: GNP has been supportive of acted like one of the transformative calculations[17]. It was utilized to programmed program age for efficient specialist practices. GNP is addressed by chart structures which comprise of three kind hubs, i.e., start hub, judgment hub and preparing hub. These hubs are associated with one another as coordinated diagram structures which give more benefits, i.e., reusability of hubs and flexibility to mostly discernible Markov choice issues. GNP has been effectively applied to the issues in unique conditions, for example, lift administrative control frameworks, stock exchanging markets and tile world (Figure 8).

FCM: Corresponding to every data point via assigning membership value to every cluster this algorithm works. On the distance between the data point and the cluster center it depends. Towards the particular cluster center gives the more is its membership when more the data is closer to cluster center. i.e, each data point's sum of membership = 1. According to formula, cluster centers are updated after performing each iteration membership. It is shown in Figure 9.

Figure 10 and 11 depict the detection rate and FAR of the proposed system while comparing it with other methods.

Figure 12a, b shows the starting snapshot of the proposed system in which we have several modules and each module contain its corresponding operation.

Figure 13a, b depicts the initializing the starting module and once the KDD dataset is loaded then a pop displaying data loaded successfully. Figure 14 depict the pre-processing stage of KDD. Figure 15 shows the PSO process after pre-processing stage. Figure 16 shows the BPN training process where the inputs are given before optimizing it and are entirely process to gain the final report. And once the training done, then a pop will be displayed. Figure 17 depict the overall view of BPN outperforming than other existing system in a graphical representation.

5. Conclusion

From this, it is clear that in today's world, how everyone gives much priority to the "security" and for that so much efficient models been released every day in much different forms. So, like that here we build an effective IDS using BPN in which system calls are collected from KDD cup 99 and are pre-processed using the sliding window and finally gave to neural network for better analysis and finally predict that, if the system behaves in normal or abnormal fashion. Also, we compared our system with other existing models under detection rate and FAR in which our system performs better with 96.5%. In future many other advance optimization techniques can be used for boosting the neural network, also other neural networks can be brought up by various researchers who can dig dive by getting inspired by this paper.

Abbreviations

DL	Deep Learning
BPN	Back Propagation Network
IDS	Intrusion Detection System
NIDS	Network Intrusion Detection System
HIDS	Host based Intrusion Detection System
DR	Detection Rate
FAR	False Alarm Rate
U2L	User to Local
R2L	Root to Local
DoS	Denial of Service
PSO	Particle Swarm Optimization
NN	Neural Network
ANFIS	Adaptive Neuro Fuzzy Inference System
F-GNP	Fuzzy Graph Neural Process
FCM	Fuzzy C Mean Clustering

Declarations

Funding:

The authors did not receive financial support from any organization for the submitted work.

Conflicts of interest/Competing interests: The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Availability of data and material:

'Not applicable', **Authors' contributions:** 'Not applicable'

Code availability: 'Not applicable', **Consent to participate:** 'Not applicable'

Ethics approval: Compliance with Ethical Standards.

Consent for publication: Authors give consent to Soft Computing Journal to publish their article.

References

1. Elmasry W, Akbulut A, Abdul Halim Zaim (2021) A Design of an Integrated Cloud-based Intrusion Detection System with Third Party Cloud Service. Open Computer Science 11(1):365–379
2. Salvatore Pontarelli G, Bianchi S, Teofili Traffic-aware Design of a High Speed FPGA Network Intrusion Detection System.Digital Object Identifier10.1109/TC.2012.105, IEEE TRANSACTIONS ON COMPUTERS.
3. Elrawy MF, Awad AI, Hesham FAH (2018) Intrusion detection systems for IoT-based smart environments: a survey. Journal of Cloud Computing 7(1):1–20
4. Thilagam T, Aruna R (2021) "Intrusion detection for network based cloud computing by custom RC-NN and optimization."ICT Express
5. Idhammad M, Afdel K, Mustapha Belouch (2018) Distributed intrusion detection system for cloud environments based on data mining techniques. Procedia Computer Science 127:35–41
6. Deshpande P, Sharma SC, Peddoju SK, Junaid S (2018) HIDS: A host based intrusion detection system for cloud computing environment. International Journal of System Assurance Engineering and Management 9(3):567–576
7. Apurva S, Patil, Dipak R Patil“ Offline host based intrusion detection based on analysis of system calls”, International Journal for Research in Engineering Application & Management (IJREAM) ISSN: 2494- 9150, Vol-02, Issue 04, July 2016
8. Ramprakash P, Sakthivadivel M, Krishnaraj N, Ramprasath J “Host-based intrusion detection system using sequence of system calls”, International Journal of Engineering and Management Research, Volume-4, Issue-2, April-2014, ISSN No.: 2250-0758
9. Sekhar R, Sasirekha K, Raja PS, Thangavel K (2021) A novel GPU based intrusion detection system using deep autoencoder with Fruitfly optimization. SN Applied Sciences 3(6):1–16
10. Wei P, Li Y, Zhang Z, Hu T, Li Z, Liu D (2019) An optimization method for intrusion detection classification model based on deep belief network. IEEE Access 7:87593–87605
11. Vinayakumar R, Alazab M, Soman KP, Poornachandran P (2019) Ameer Al-Nemrat, and Sitalakshmi Venkatraman. "Deep learning approach for intelligent intrusion detection system. IEEE Access 7:41525–41550
12. Kim J, Kim J, Kim H, Shim M, Choi E (2020) "CNN-based network intrusion detection against denial-of-service attacks." *Electronics* 9, no. 6 : 916
13. Amudha P, Karthik S, Sivakumari S (2015) "A hybrid swarm intelligence algorithm for intrusion detection using significant features." *The Scientific World Journal* (2015)
14. Staudemeyer RC (2015) Applying long short-term memory recurrent neural networks to intrusion detection. South African Computer Journal 56(1):136–154
15. Huang X (2021) "Network Intrusion Detection Based on an Improved Long-Short-Term Memory Model in Combination with Multiple Spatiotemporal Structures." *Wireless Communications and Mobile Computing* (2021)

16. Tan X, Su S, Zuo Z, Guo X, Sun X (2019) "Intrusion detection of UAVs based on the deep belief network optimized by PSO." *Sensors* 19, no. 24 : 5529
17. Azad C (2017) "Fuzzy min–max neural network and particle swarm optimization based intrusion detection system. Microsyst Technol 23(4):907–918
18. Venkatraman S, Alazab M (2018) "Use of data visualisation for zero-day malware detection." *Security and Communication Networks* (2018)
19. Kunhare N, Tiwari R, Dhar J (2020) "Particle swarm optimization and feature selection for intrusion detection system." *Sādhanā* 45, no. 1 : 1-14
20. Liu J, Yang D, Lian M, Li M (2021) Research on Intrusion Detection Based on Particle Swarm Optimization in IoT. IEEE Access 9:38254–38268
21. Solanki M, Dhamdhere V (2015) ”Intrusion detection system using means of data mining by using c 4.5 algorithm”,International Journal of Application or Innovation in Engineering & Management, Volume 4, Issue 5,
22. Yin C, Zhu Y, Fei J, He X (2017) A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks. IEEE Access 5:21954–21961

Figures

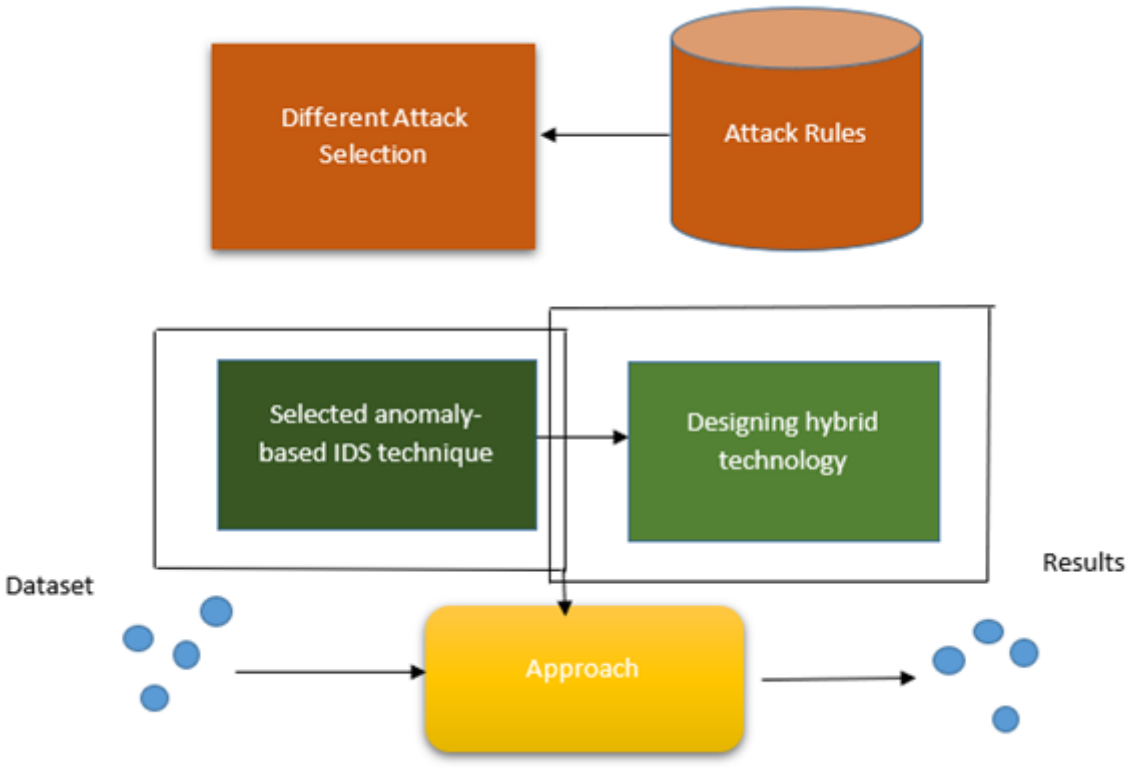


Figure 1

IDS using DL basic flow

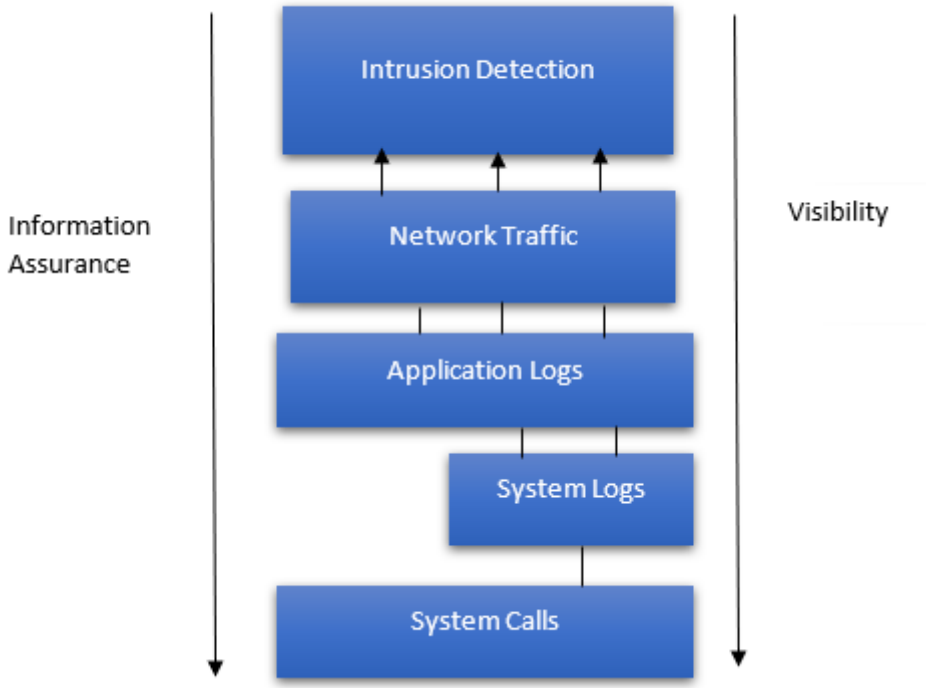


Figure 2

Types of IDS in simple network system

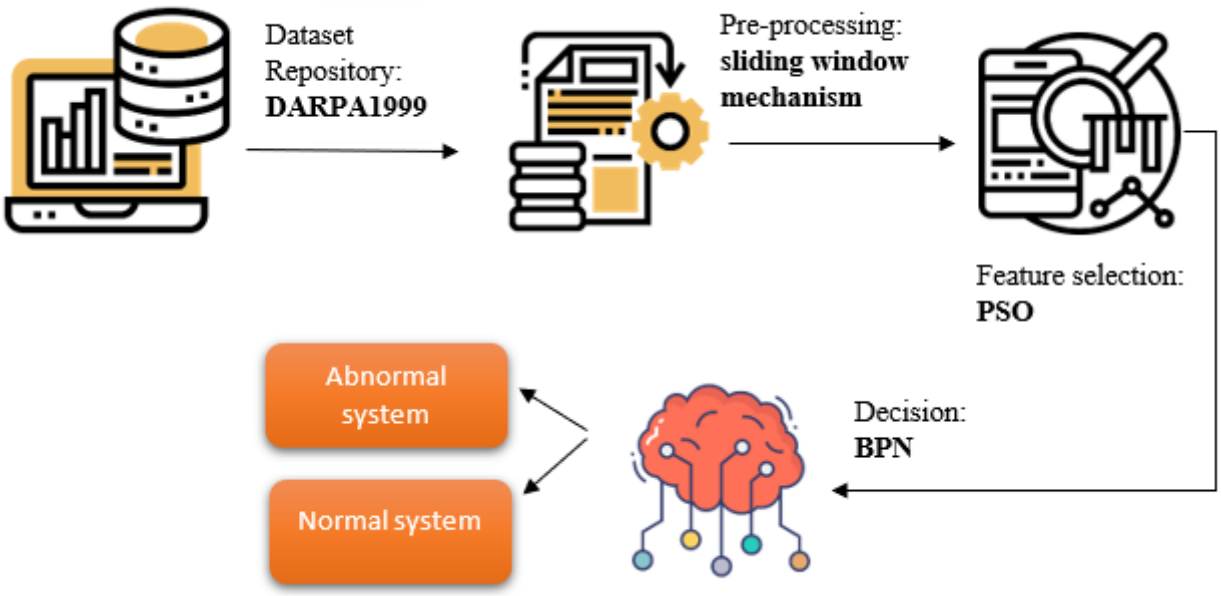


Figure 3

Block diagram of proposed IDS using BPN

position 3	position 2	position 1	current
			execve
		execve	brk
fstat	execve, mmap	brk, close	open
execve	brk	open	fstat
brk, mmap	open, close	fstat, open	mmap
open	fstat	mmap	close
close	open	mmap	munmap

Figure 4

Sample system call sequence

Figure 5

BPN

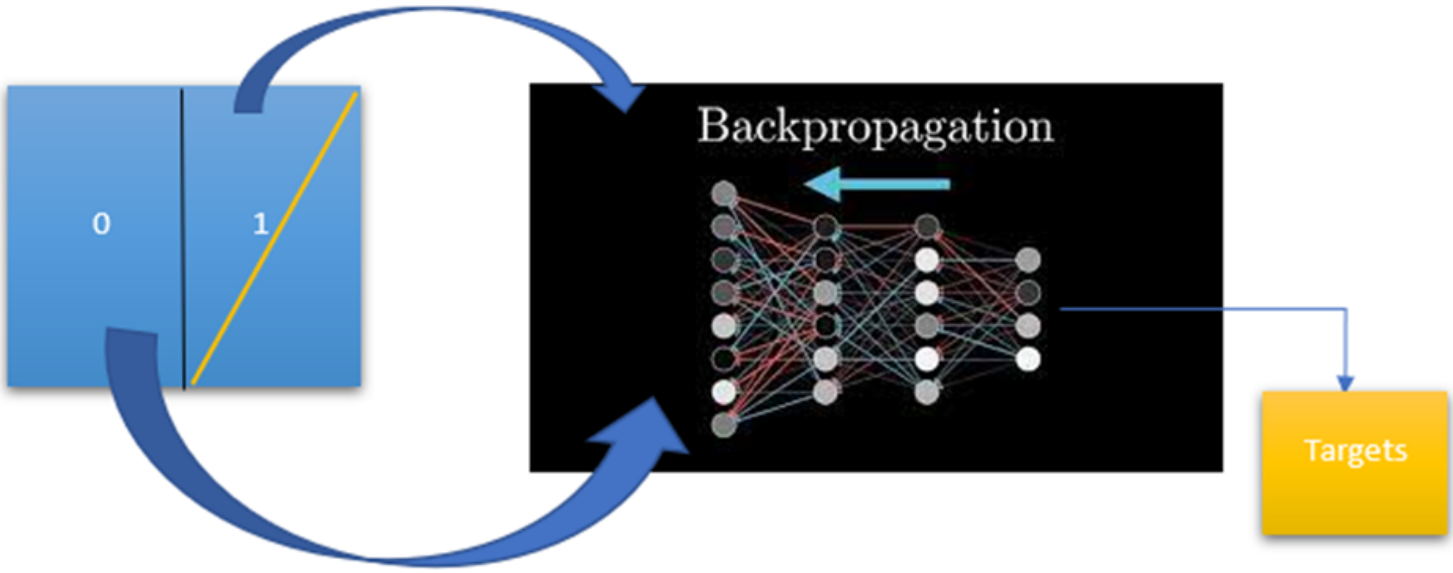


Figure 6

Training of BPN

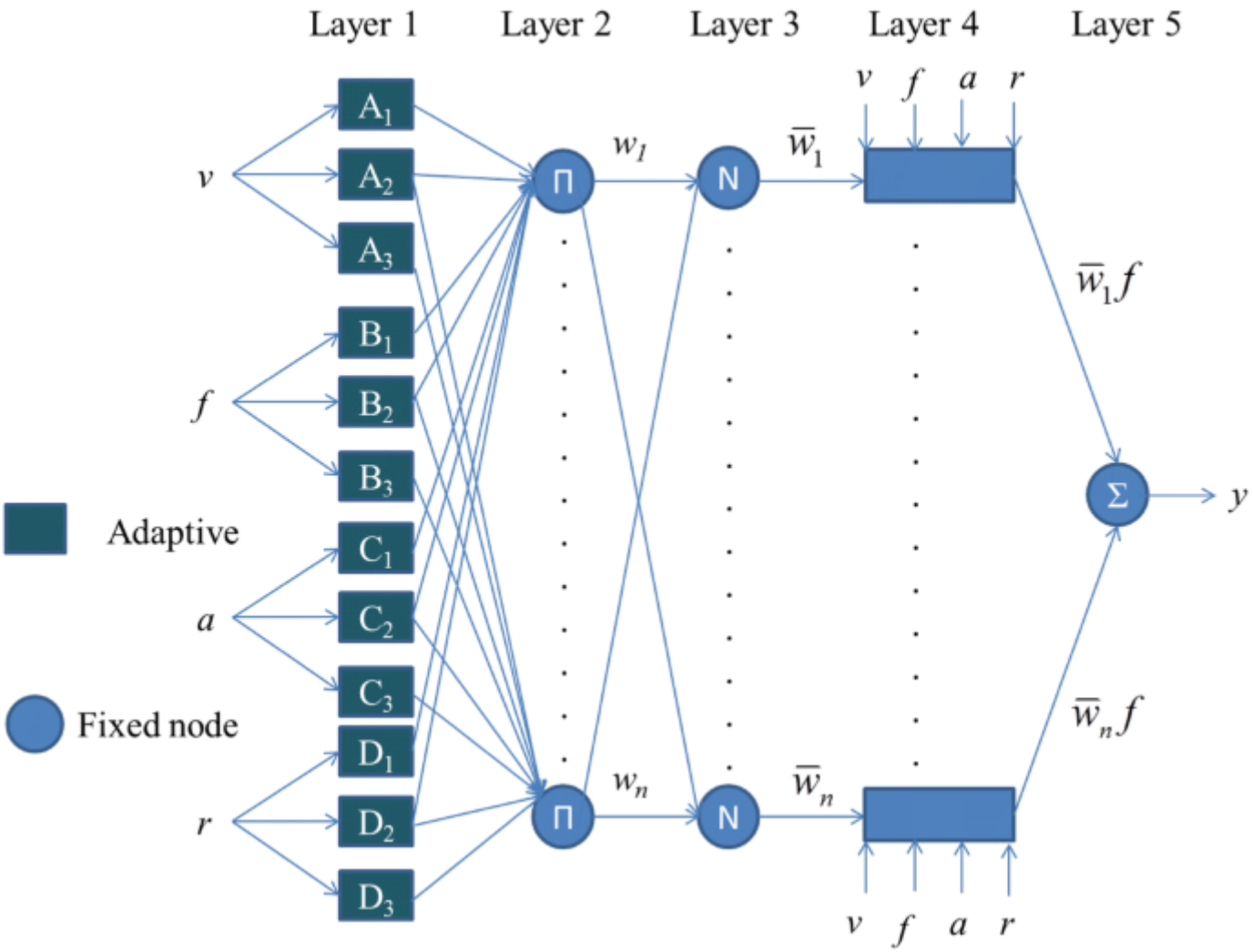


Figure 7

ANFIS design

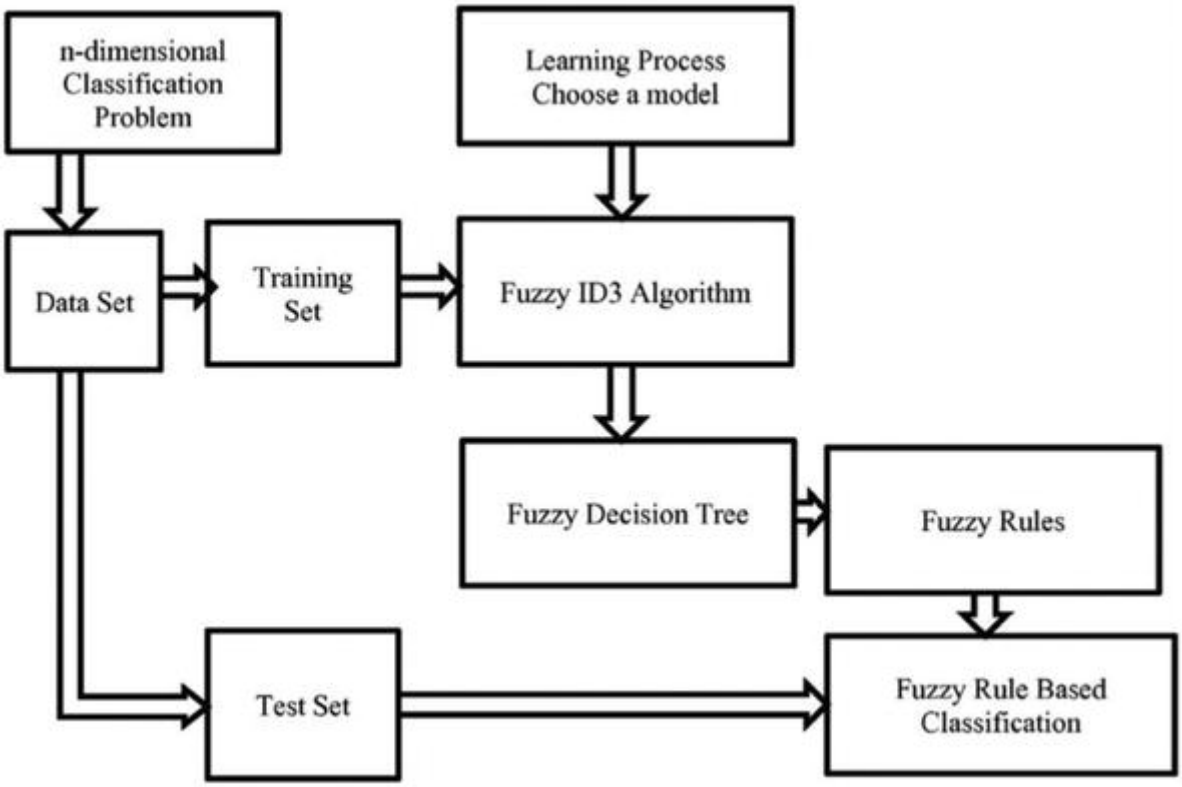


Figure 8

F-GNP architecture

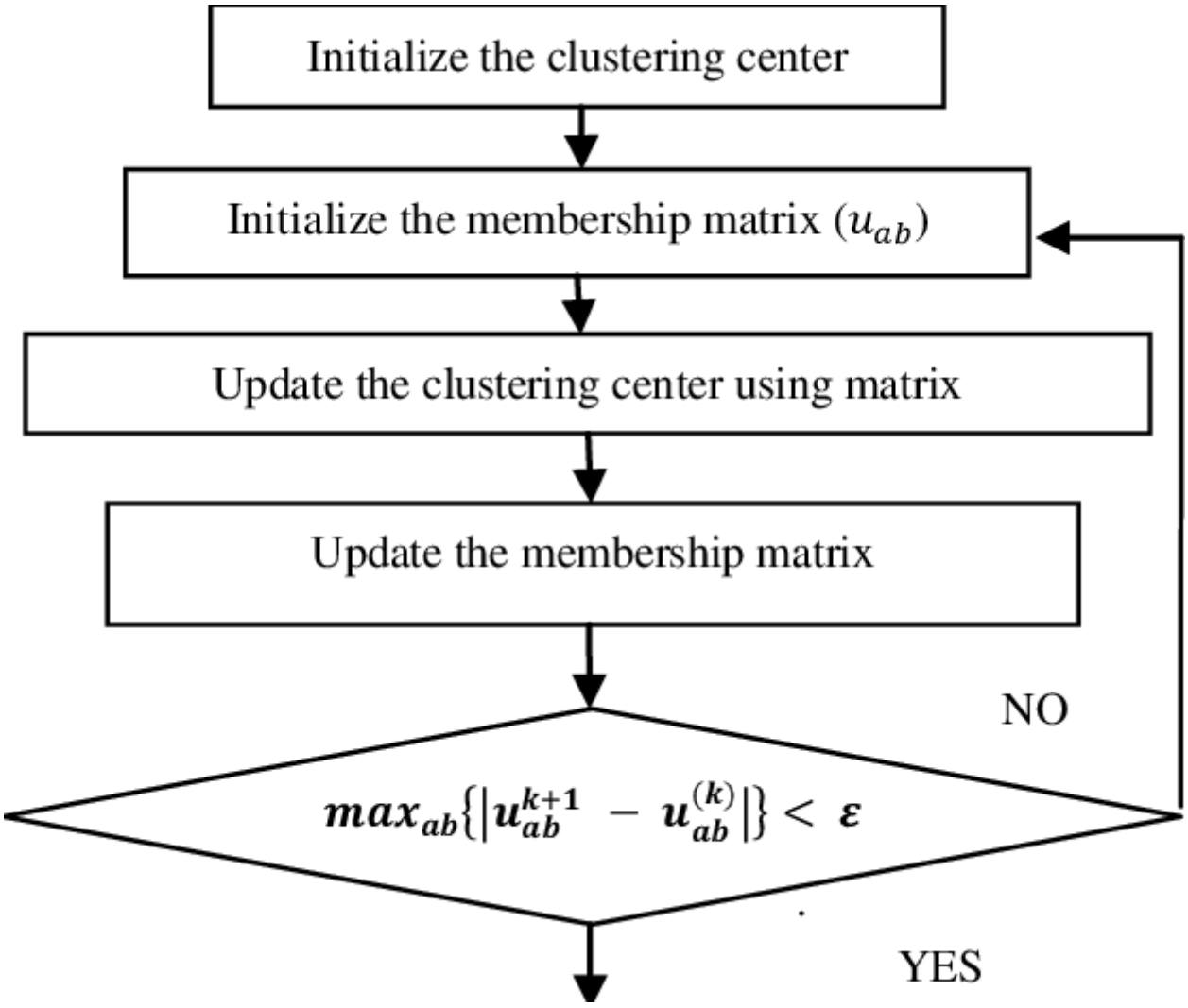


Figure 9

FCM design

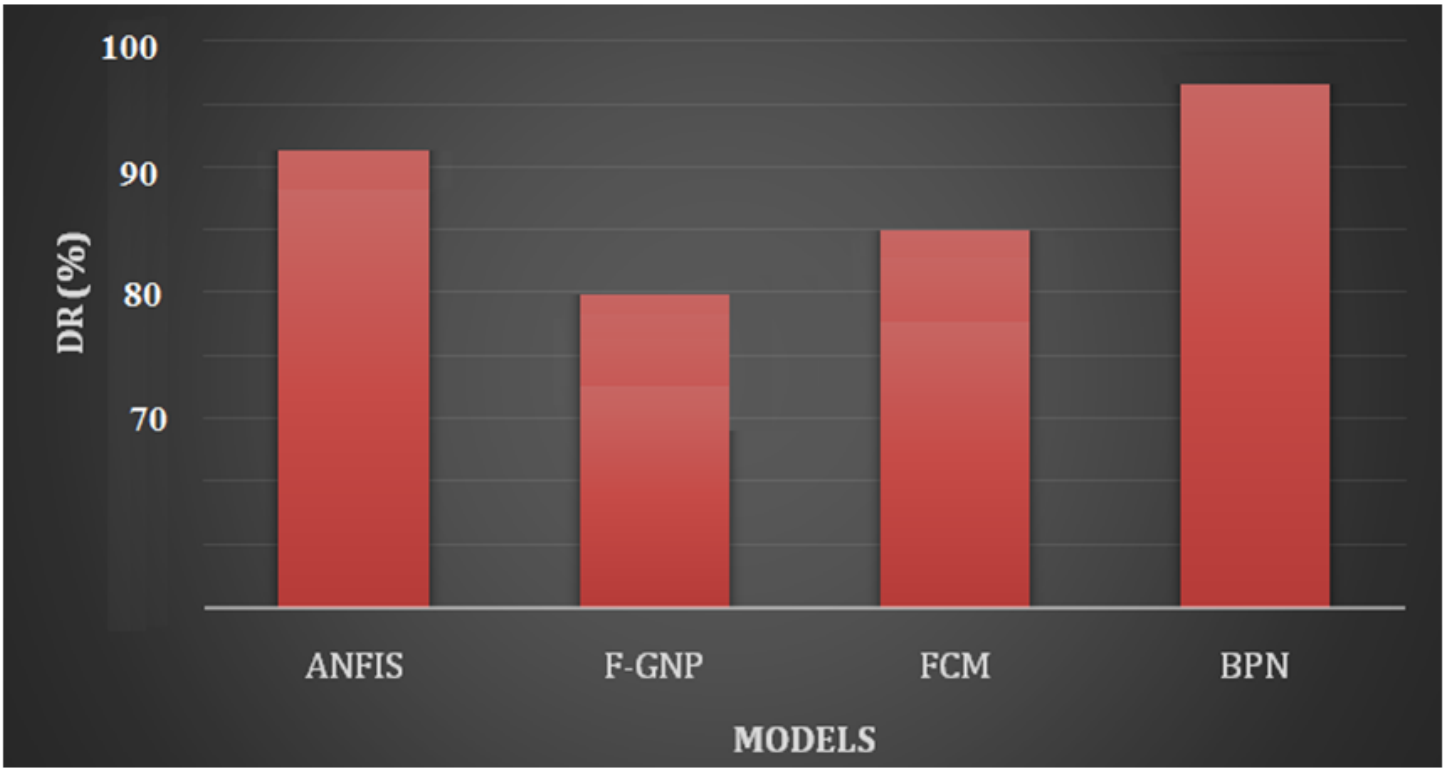


Figure 10

Comparative analysis of various models under DR

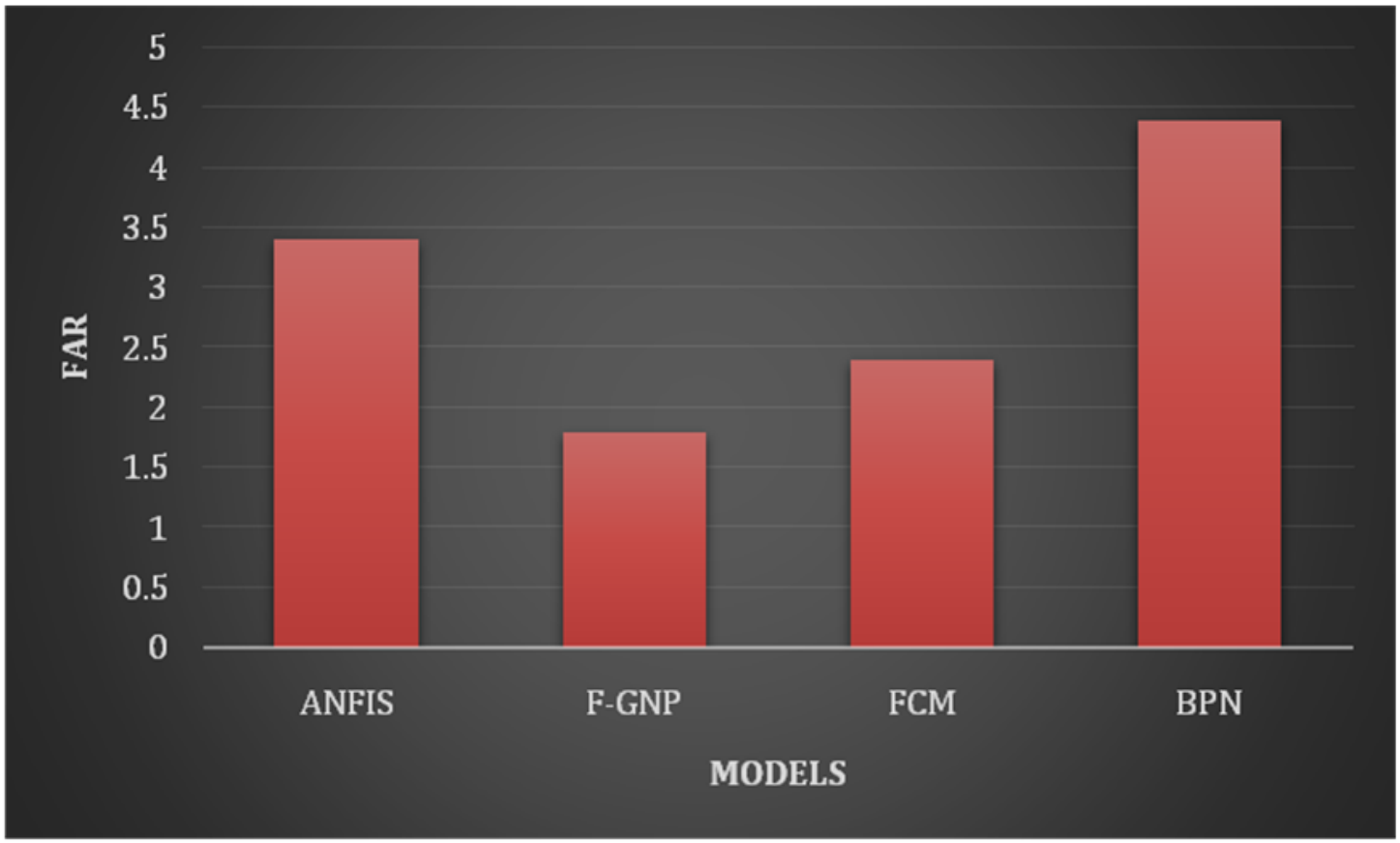


Figure 11

Comparative analysis of various models under FAR

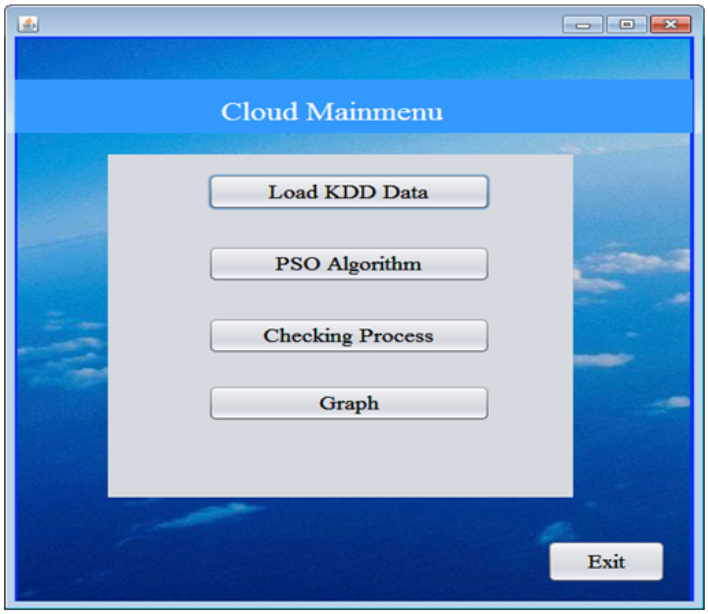
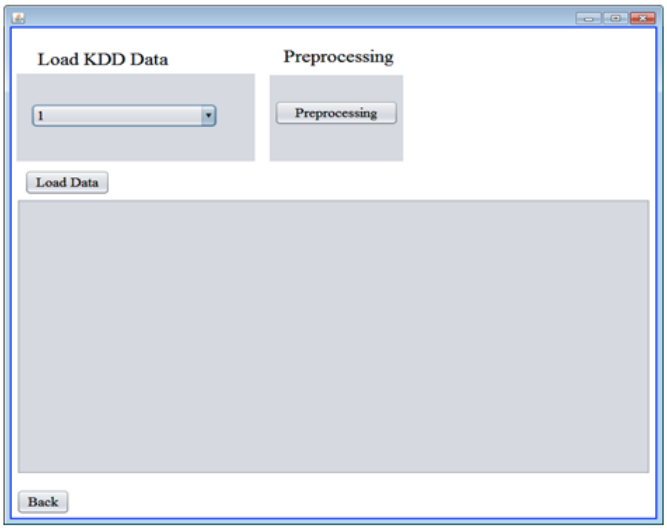


Figure 12

a. starting process of the model, b) the process of the model



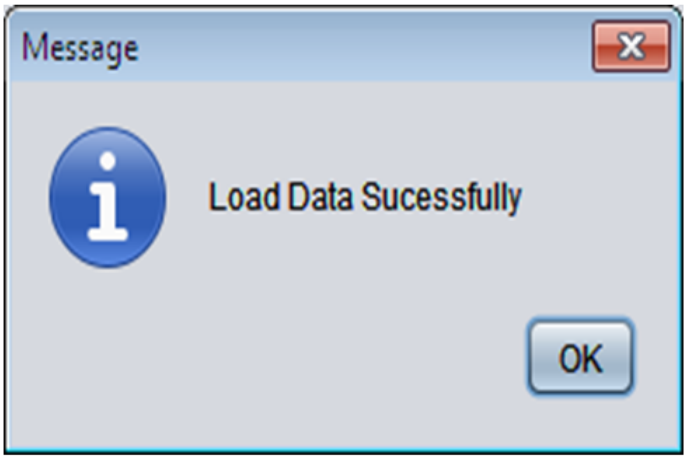
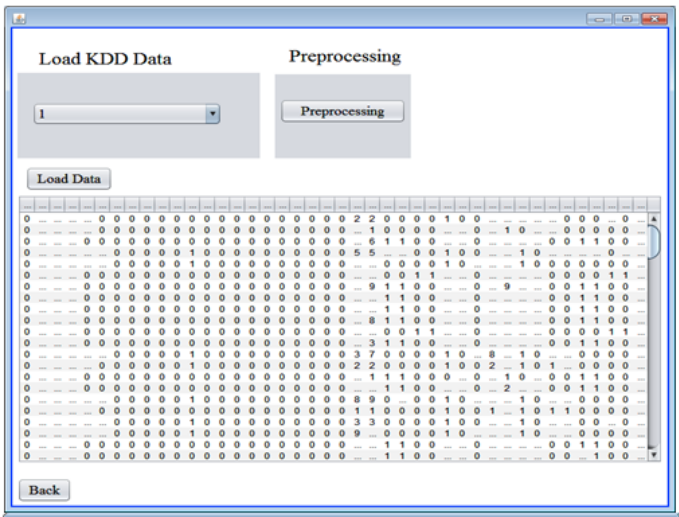
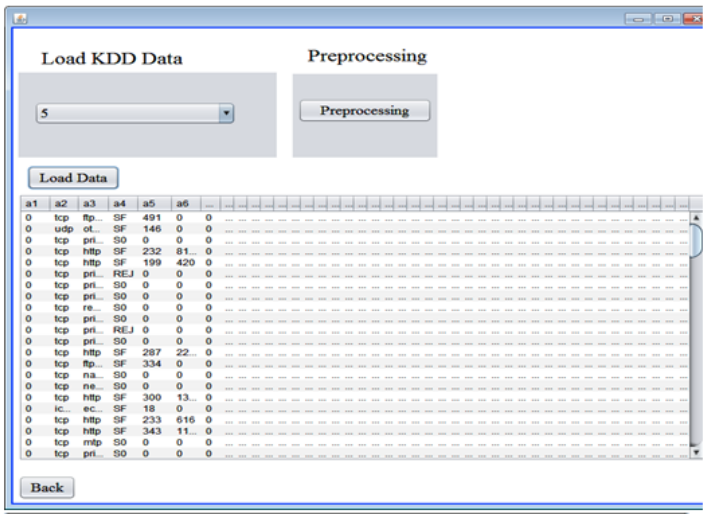
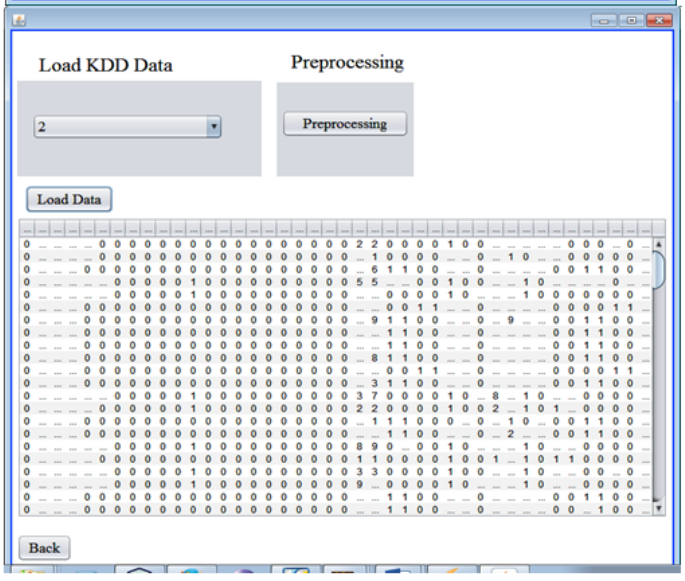


Figure 13

a. Initializing the first process, b) once the dataset is loaded, pop up will displayed







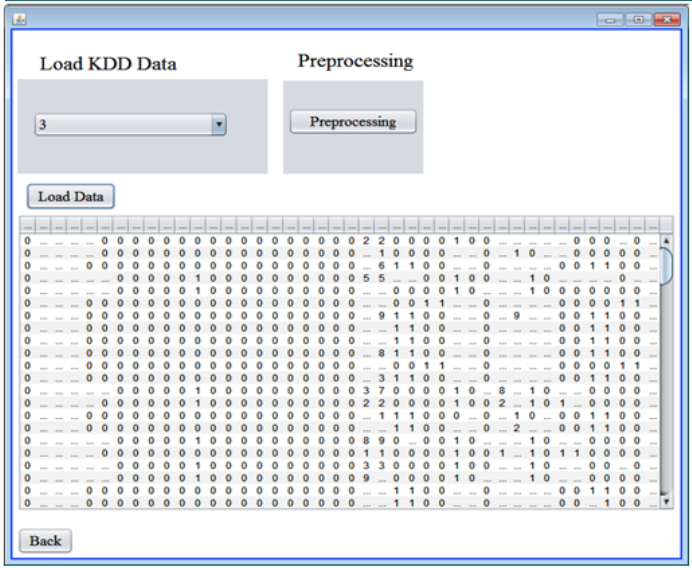


Figure 14

Pre-processing snapshot of KDD dataset

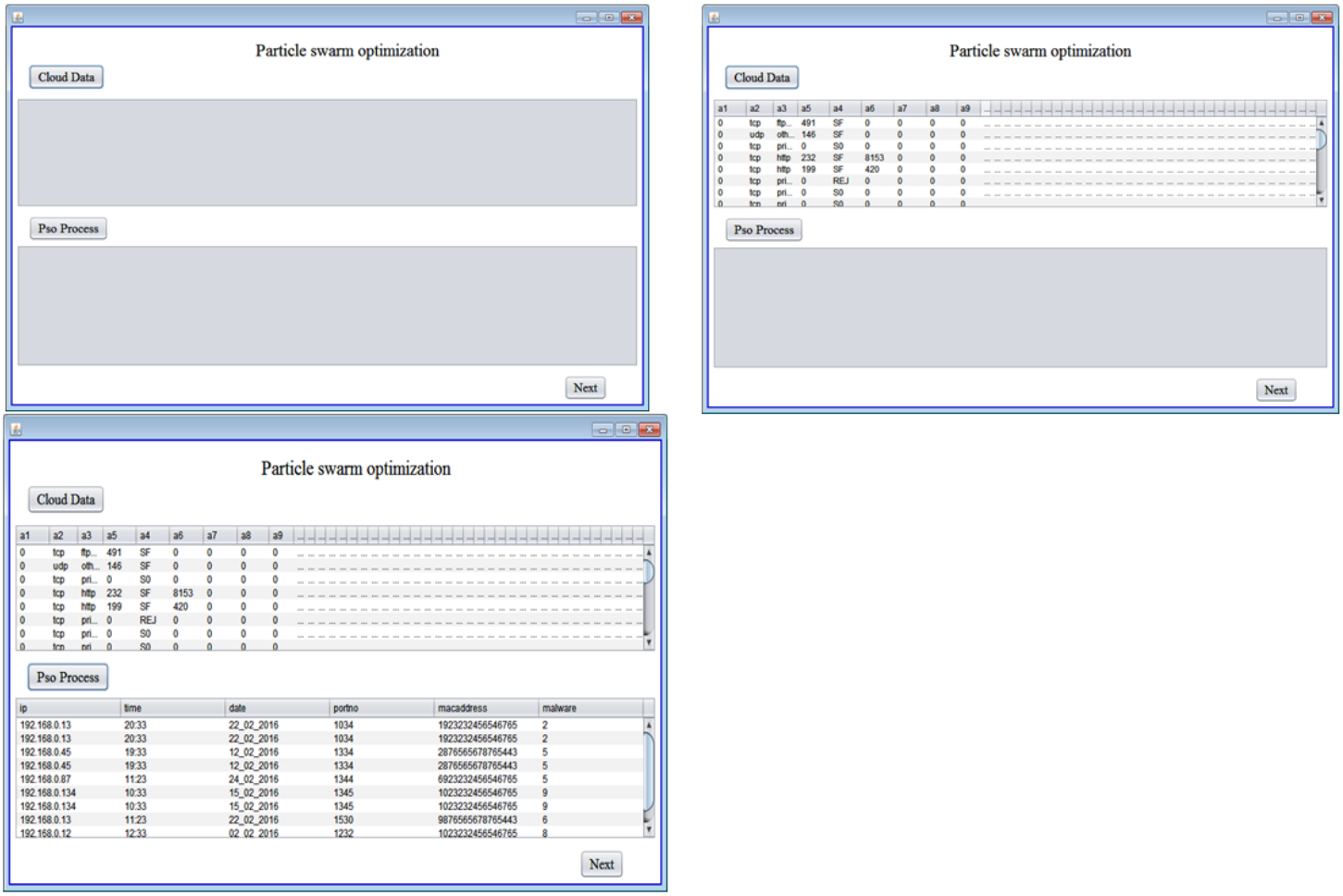


Figure 15

PSO process after the pre-processing stage

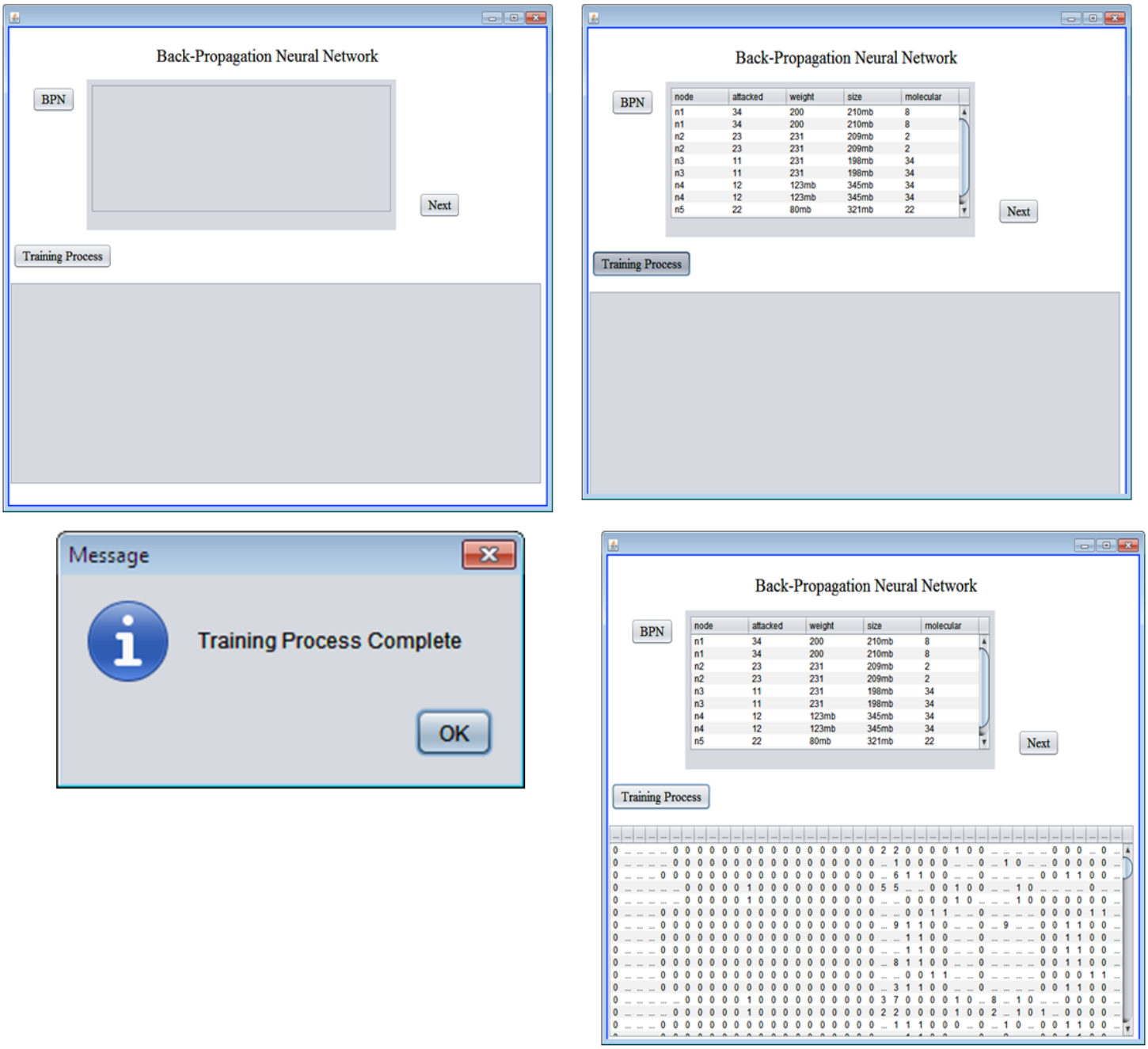


Figure 16

Snapshot showing BPN process after optimization is done

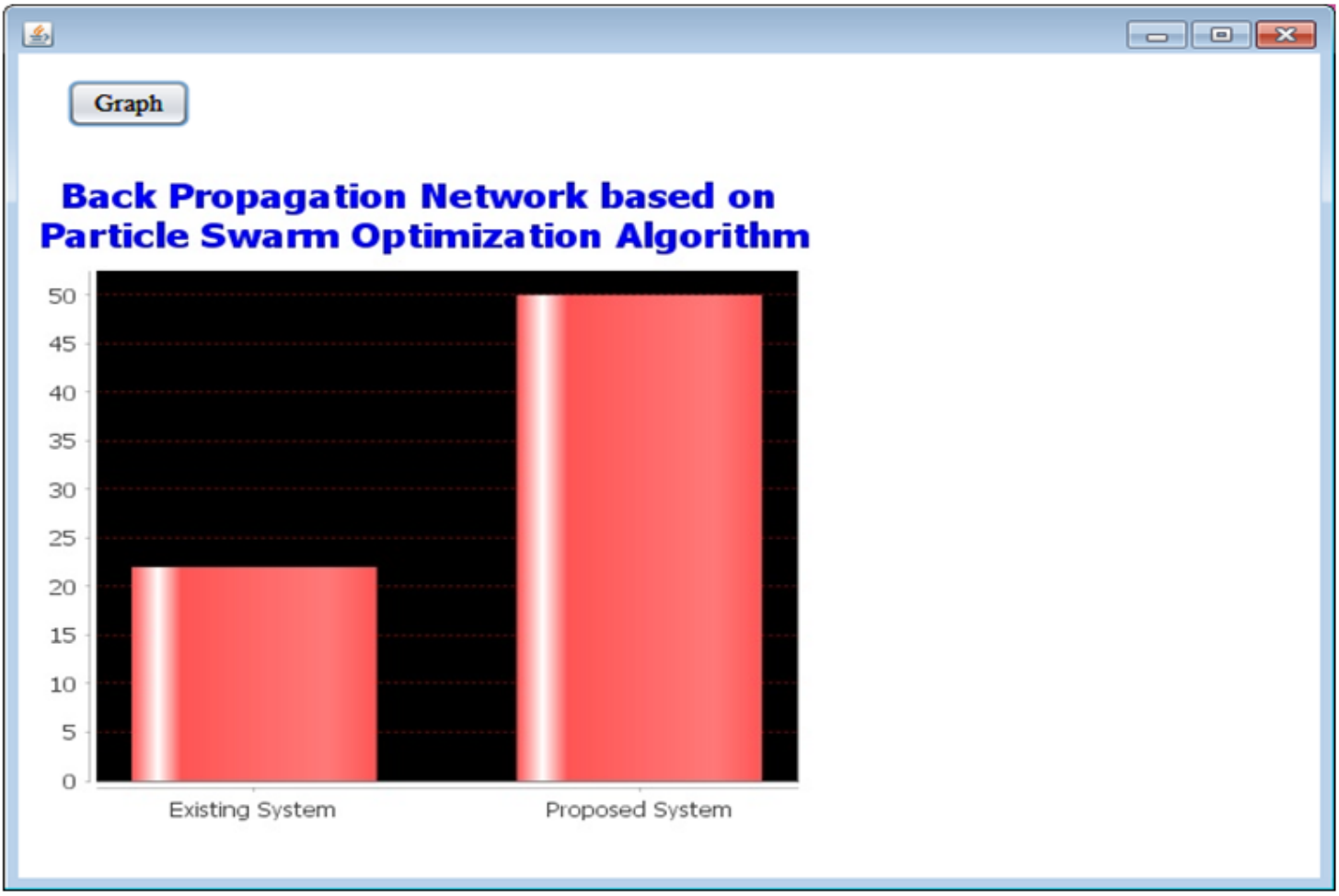


Figure 17

Comparison of existing and proposed system in overall fashion