ISSN:0975-9646

Classifier Model for Intrusion Detection Using Bio-inspired Metaheuristic Approach

P.Amudha[#1], S.Karthik[*2], S.Sivakumari[#3]

[#] *Department of CSE, Faculty of Engineering, Avinashilingam University Coimbatore, Tamilnadu, India*

[*] *Department of CSE, SNS College of Technology, Coimbatore, Tamilnadu, India.*

***Abstract*— In machine learning and statistics, feature selection is the technique of selecting a subset of relevant features for building robust learning models. In this paper we propose a bio-inspired BAT algorithm as feature selection method to find the optimal features from the KDDCup'99 intrusion detection dataset obtained from UCI Machine Learning repository. Neural Networks (NN) as a classifier collects data randomly from the dataset and constructs a training dataset with original records using bagging approach. The performance of Neural Network with Repeated Incremental Pruning to Produce Error Reduction (RIPPER) is compared with the Neural Network with C4.5 decision tree and the experimental result shows that the Neural Network with RIPPER outperforms the other algorithm.**

***Keywords*— Bio-inspired, RIPPER, bagging, decision tree, KDDCup'99..**

I. INTRODUCTION

Feature selection aims to find the most important information from a given set of features and helps to have better understanding about the data, important features and relationship with each other. Feature selection improves the overall accuracy, reduces the number of false alarms and improves the detection rate of instances in the training data. A feature selection algorithm is a blend of search methods which proposes new feature subsets. Recently, bio-inspired approaches such as swarm intelligence are used to solve real-time complicated problems. The swarm intelligence based algorithms inspired by the behaviour of animals has been successfully applied to optimization, robotics and military applications [1]. This paper aims to analyse the effect of bat algorithm as feature selection method on intrusion detection dataset and to study its effect on the ensemble classifiers.

II. BACKGROUND STUDY

The wrapper method was first implemented by R.Kohavi and G.H.John [2] for feature selection in machine learning. Feature selection keeps the original features as such and selects subset of features that predicts the target class variable with maximum classification accuracy. Srinivas Mukkamala and Sung [3], Ivor W. Tsang et al.,[4] S.Chebrolu et al.,[5], Y. Chen et al. [6], T.P.Fries [7] and K. Makkithaya et al.,[8] adopted AI-based feature selection techniques for intrusion detection.

Evolutionary computation and swarm intelligence techniques are great examples that nature has been a continuous source of inspiration. The behaviour of bees, ants, glow-worms, fireflies, fishes and other organisms have inspired swarm intelligence researchers to devise new optimization algorithms [3]. Xin-She Yang proposed a new meta-heuristic method, Bat Algorithm, based on the echolocation behaviour of bats [9][10]. R.S. Parpinelli, H.S.Lopes discussed on swarm intelligence for growing complexity of real-world problems which has motivated computer scientists to search for efficient problem-solving methods [1].

R.Y.M.Nakamura et al. applied bat algorithm as feature selection technique [11]. Amir Hossein Gandomi, Xin-She Yang [12] applied Bat algorithm to solve constraint optimization tasks. The optimal solution obtained by bat algorithm was found to be better than the best solutions provided by other methods. Koffka Khan, Ashok Sahai compared BA, GA, PSO, BP and LM for Training Feed forward Neural Networks intended to show the superiority of the new metaheuristic bat algorithm (BA) over other more standard algorithms in neural network training [13].

Mathew Miller discussed on learning Cost-Sensitive Classification Rules for Network Intrusion Detection using RIPPER [14]. RIPPER was introduced by Cohen [15] as a successor of the IREP algorithm for rule induction (Fürnkranz and Widmer [16]). RIPPER offers a number of modifications to IREP, C4.5, and C4.5 rules which have proven to yield faster training times and lower error rates. Komviriyavut, Sangkatsanee, Charnsripinyo[17] presented two network intrusion detection (IDS) techniques which are C4.5 decision tree and Ripper rules to assess and test an online dataset(RLD09 dataset).

III. METHODOLOGY

A. Bat Algorithm

Bat Algorithm is a relatively new population based bio-inspired approach based on hunting behaviour of bats. Bats are fascinating animals and they are the only mammals with wings and which have advanced capability of echolocation to detect prey, avoid obstacles and locate their roosting crevices in the dark. These bats emit a very loud sound pulse and listen for the echo that bounces back from the surrounding objects. Their pulses vary in properties and can be correlated with their hunting strategies, depending on the species. Most bats use short, frequency-modulated signals to sweep through about an octave, while others more often use constant-frequency signals for echolocation. Their signal bandwidth varies depends on the species, and often increased by using more harmonics.

Inspired by the behaviour of bats, Yang [9][10] has developed a new meta-heuristic optimization technique called Bat Algorithm and has idealized some rules:

1. All bats use echolocation to sense distance, and they also 'know' the difference between food/prey and background barriers in some magical way;
2. Bats fly randomly with velocity v_i at position x_i with a fixed frequency $f_{\min}$, varying wavelength λ and loudness A_0 to search for prey. They can automatically adjust the wavelength (or frequency) of their emitted pulses and adjust the rate of pulse emission r $\in$ [0, 1], depending on the proximity of their target;
3. Although the loudness can vary in many ways, we assume that the loudness varies from a large (positive) A_0 to a minimum constant value A_{min}.

The implementation of BA is more complicated than many other meta-heuristic algorithms [1] because each agent (bat) is assigned a set of interacting parameters such as position, velocity, pulse rate, loudness, and frequencies. This interaction affects the quality of a solution and the time needed to obtain such solution.

The algorithm consists of the following components:

- initialization of parameters
- generation of new solutions
- local search
- generation of a new solution by flying randomly
- find the current best solution

B. Feature Selection Using Bat Algorithm

In this section, we present bio-inspired feature selection approach using bat algorithm to find the best combination of features. Bat algorithm can deal with problem of the high dimensionality and finding the most informative features in a search space. The algorithm has a capability of automatically zooming into a region where promising solutions have been found which accompanied by the automatic switch from explorative moves to local intensive exploitation. As a result, it has a quick convergence rate, at least at early stages of the iterations compared with other algorithms.

Initially, each bat is randomly assigned a frequency which is drawn uniformly from [$f_{\min}$, $f_{\max}$]. For the local search part, once a solution is selected among the current best solutions, a new solution for each bat will be generated using equation (4).

$$x_{new} = x_{old} + \in A^t \quad (4)$$

where, $\varepsilon \in$ [-1, 1] is a random number, while $A^T = A_i^T$ is the average loudness of all the bats at this time step. The loudness A_i and the rate r_i of pulse emission have to be updated accordingly as the iterations proceed.

The bat algorithm [9][10] is given below:

Objective function $f(x)$, $x=(x_1,....x_d)^T$

Initialize the bat population x_i =(i = 1,2…n) and v_i

Define Pulse frequency f_i at x_i

Initialize the rates r_i and the loudness A_i

While (t < Max number of iterations)

Generate new solutions by adjusting frequency, updating velocities and locations/solutions

$$f_{i=} f_{\min} + (f_{\max} - f_{\min})\beta \quad (1)$$

$$v_i^t = v_i^{t-1} + (x_i^t - x^*) f_i \quad (2)$$

$$x_i^t = x_i^{t-1} + v_i^t \quad (3)$$

If (rand > r_i) then

Select a solution among the best solutions

Generate a local solution around the selected best solution

End if

Generate a new solution by flying randomly

If (rand < A_i & $f(x_i) < f(x_*)$) then

Accept the new solutions

Increase r_i and reduce A_i

end if

Rank the bats and find the current best x_*

end while

Post process results and visualization

B. Classification Using RIPPER

A rule-based classifier is a method for classifying examples using if ... *then* ... rules. On many problems rule learning systems outperform decision tree learners and one of the active techniques is Repeated Incremental Pruning to Produce Error Reduction (RIPPER). RIPPER is an optimized version of IREP. In REP algorithms, the training data is split into a growing set and a pruning set. Using some heuristic method, an initial rule set is formed and by applying pruning operators, the rule set is simplified. At each stage of simplification, the pruning operator chosen is the one that yields the greatest reduction of error on the pruning set. Simplification ends when applying any pruning operator would increase error on the pruning set[17].

A single RIPPER rule is of the form $r = <r_A, r_C>$, consisting of a premise part r_A and a consequent part r_C. The premise part r_A is a conjunction of predicates (selectors) which are of the form ($A = v_i$) for nominal and ($A_i \ \theta \ v_i$) for numerical attributes, where $\theta \in \{\leq, \approx, \geq\}$ and $v_i \in D_i$. The consequent part r_C is a class assignment of the form (class = l). A rule $r = \langle r_A, r_C \rangle$ is said to *cover* an instance $x = (x_1.....x_n)$ if the attribute values x_i satisfy all the predicates in r_A.

Each individual rule is learned in two steps. The training data, which has not yet been covered by any rule, is therefore split into a growing and a pruning set. In the first step, the rule will be specialized by adding antecedents which were learned using the growing set. Afterward, the rule will be generalized by removing antecedents using the pruning set. When RIPPER learns a rule for a given class, the examples of that class are denoted as positive instances, whereas the examples from the remaining classes are denoted as negative instances [18]-[20].

C. Enhanced RIPPER Using Neural Network Ensemble

Neural Network (NN) consists of input layer with number of input nodes equal to the number of selected features and the output layer based on number classes used. The output of each node is propagated from the input layer through the hidden layer to the output layer. The output node with the highest value is chosen and its corresponding class is named as its output. As neural network and decision tree are unstable classifiers bagging approach is employed. Bagging creates a training dataset by sampling with replacement 'n' times from the dataset containing 'n' records.

The trained neural network ensemble is employed to generate a new training data by replacing the desired class labels of the original training data, with those output from the trained NN ensemble. The network pattern is identified based on the predicted output from each of the NN using voting algorithm. Voting algorithm chooses the class label receiving most number of votes as the final output of the ensemble.

In Learning process, the training data of the neural network is sorted by class labels in the ascending order and the corresponding class frequencies also defined. In that Rules are then learned for the first m-1 classes, starting with the smallest one. Once the rules are generated from the training data in the neural network, then the instances covered by that rule are removed from the training set in the neural network. Then process continues until all the training samples or training data creates the rules. The algorithm then proceeds with the next class. Finally, when RIPPER finds no more rules to learn, a default rule (with empty antecedent) is added for the last (and hence most frequent) class. In this algorithm, rules are added based on the information gain measure.

IV. DATASET DESCRIPTION

In this section we discuss the intrusion detection dataset KDDCup'99[21], which was derived from the 1998 DARPA Intrusion detection Evaluation program prepared and managed

by MIT Lincoln Laboratory. The dataset is a collection of simulated raw TCP dump data over a period of nine weeks for a LAN simulating a typical U.S. Air Force LAN. There are 4,898,430 labeled and 311,029 unlabeled connection records in the dataset. The labeled connection records consist of 41 attributes. . The detail of attacks of labeled records of KDDCup'99 dataset is given in Table I.

TABLE I
DETAILS OF ATTACKS OF LABELLED RECORDS

Category of attack	Attack Name
Normal	Normal
DoS	Neptune, Smurf, Pod, Teardrop, Land, back
Probe	Portsweep, Ipsweep, Nmap, satan
U2R	Bufferoverflow, LoadModule, Perl, Rootkit
R2L	Guesspassword, Ftpwrite, Imap, Phf, Multihop, Warezmaster, Warezclient

The complete listing of the set of features in the dataset is given in Table II, where c and s represents continuous and symbolic respectively.

TABLE III
SET OF FEATURES OF KDDCUP'99 DATASET

F No.	Name of the feature	F No.	Name of the feature
1	duration (c)	22	is_guest_login (s)
2	protocol_type (s)	23	count (c)
3	service (s)	24	srv_count (c)
4	flag (s)	25	serror_rate (c)
5	src_bytes (c)	26	srv_serror_rate (c)
6	dst_bytes (c)	27	rerror_rate (c)
7	land (s)	28	srv_serror_rate (c)
8	wrong_fragment (c)	29	same_srv_rate (c)
9	urgent (c)	30	diff_srv_rate (c)
10	hot (c)	31	srv_diff_host_rate (c)
11	num_failed_logins (c)	32	dst_host_count (c)
12	logged_in (s)	33	dst_host_srv_count (c)
13	num_compromised (c)	34	dst_host_same_srv_rate (c)
14	root_shell (c)	35	dst_host_diff_srv-rate (c)
15	su_attempted (c)	36	dst_host_same_srv_port_ rate (c)
16	num_root (c)	37	dst_host_srv_diff_host_ rate (c)
17	num_file_creations (c)	38	dst_host_serror_rate (c)
18	num_shells (c)	39	dst_host_srv_serror_rate (c)
19	num_access_files (c)	40	dst_host_rerror_rate (c)
20	num_outbound_cmd (c)	41	dst_host_srv_rerror_rate (c)
21	is_host_login (s)		

The dataset consists of one type of normal data and 22 different attack types categorized into 4 classes namely: *Denial of Service* (DoS), *Probe*, *User-to-Root* (U2R), *Remote-to-Login* (R2L).

- *Denial of Service (DoS):* Attacker tries to prevent legitimate users from using a service.
- *Probe:* Attacker tries to gain information about the target host.
- *User-to-Root* (*U2R)*: Attacker has local access to victim machine and tries to gain super user privileges.
- *Remote-to-Login (R2L):* Attackers does not have an account on the victim machine. Hence tries to gain access.

V. Results And Discussions

A. Experimental Setup

We performed our experiment using 10% of the overall KDD Cup'99 labelled dataset which contains 4, 94,020 records having 41 features. The distribution of connections types in KDD99 10% training dataset is given in Table III. One of the most important deficiencies in the KDDCup dataset is the large number of redundant instances, which causes the learning algorithms to be biased towards the frequent instances and prevent learning from infrequent instances which are harmful to networks. Hence duplicate instances are removed and selected random sample of 10% normal data and 10% Neptune attack in DoS class. They include 8783 normal instances, 7935 DoS instances, 2131 Probe instances, 52 U2R instances and 999 R2L instances. Four new sets of data are generated with the normal class and four categories of attack (*DoS*, *Probe*, *U2R*, and *R2L*). In each data set, instances with the same attack category and 10% normal instances are included, where each dataset has its own distribution of categories of instances [22].

TABLE IIIII
DISTRIBUTION OF CONNECTION TYPES IN KDDCUP99 10% TRAINING DATASET

Class	Number of records	Percentage of occurrence
Normal	97,277	19.69
DoS	391,458	79.24
Probe	4,107	0.83
U2R	52	0.01
R2L	1,126	0.23
Total	494,020	100.00

The performance metrics like accuracy, detection rate, false alarm rate, precision are recorded for the classification algorithm and the formulae to calculate is given in equations (5) to (8):

$$\text{Accuracy} = (TP + TN) / (TP + FP + TN + FN) \quad (5)$$

$$\text{Detection Rate} = TP / (TP + FN) \quad (6)$$

$$\text{False Alarm Rate} = FP / (TN + FP) \quad (7)$$

$$\text{Precision} = TP / (TP + FP) \quad (8)$$

- True Positive (TP) is the number of attacks correctly classified
- True Negative (TN) is the number of normal records correctly classified
- False Positive (FP) is the number of normal records incorrectly classified
- False Negative (FN) is the number of attacks incorrectly classified

The composition of dataset and number of features selected using the bio-inspired bat algorithm is shown in Table IV. The experimental result for the neural network with RIPPER is shown in Table V.

TABLE IV
COMPOSITION OF DATASET USING BAT ALGORITHM

Dataset	No. of	No. of reduced
Normal+DoS	16718	14
Normal+Probe	10914	17
Normal+ U2R	8835	17
Normal+R2L	10782	16

Fig. 1 shows the graphical representation of accuracy of the classifiers on all datasets. The results show that neural network with RIPPER have higher accuracy of 95.15% on DoS+10%Normal dataset when compared to the other datasets. Also it is observed that neural network with RIPPER outperforms the other method in terms of accuracy (95.91% in Dos+10%Normal, 94.59% in Probe+10%normal dataset, 94.60% in R2L+10%normal and 94.70% U2R+10%normal).

Fig. 2 shows that the hybridization of neural network with RIPPER on DoS+10%Normal dataset have the highest detection rate of 95.15% when compared to the other dataset and it obtains high detection rate than decision tree. The graphical representation of False alarm rate is given in Fig. 3. It is shown that the neural network with RIPPER on Probe+10%Normal dataset have the lowest false alarm rate of 0.0484% when compared to the other datasets. Compared to decision tree approach, false alarm rate is lower for neural network with RIPPER (0.0484% in Dos +10%Normal, 0.0540% in Probe+10%normal dataset, 0.0529% in R2L+10%normal and 0.0539% in U2R+10%normal).

Fig. 4 shows that the neural network with RIPPER on probe+10%Normal and U2R+10% normal datasets have the highest precision of 96.98% when compared to the other datasets and is also higher on all datasets when compared to decision tree method (96.52% in Dos+10%Normal, 96.98% in Probe+10%Normal dataset, 96.89% in R2L+10%Normal and 96.98% U2R+10%Normal).

TABLE V
EXPERIMENTAL ANALYSIS OF NN-RIPPER

Dataset	Accuracy (in %)	Detection Rate (in %)	False Alarm rate(in %)	Precision (in %)
DoS+10%Normal	95.15	95.15	0.0484	96.33
Probe+10%Normal	94.59	94.59	0.0540	96.89
U2R+10%Normal	94.70	94.70	0.0529	96.79
DoS+10%Normal	94.60	94.60	0.0539	96.87

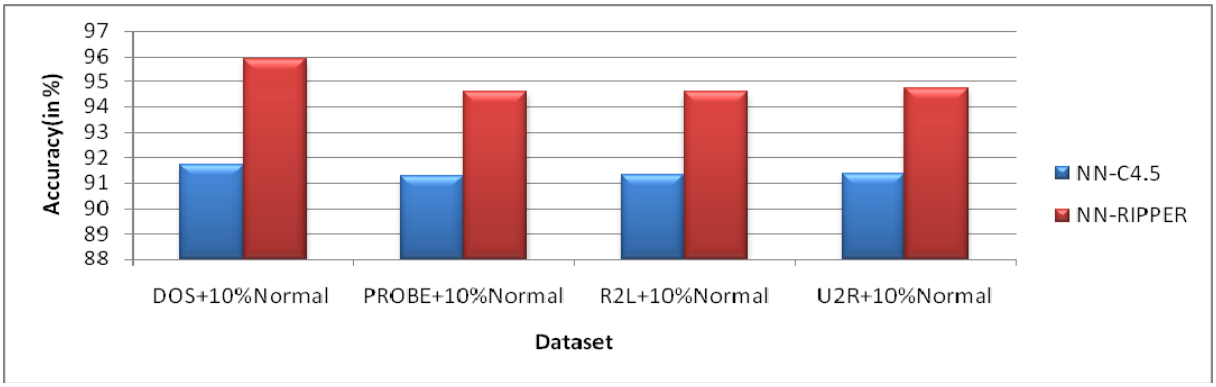

Fig. 1 Accuracy Comparison

Fig. 2 Detection rate comparison

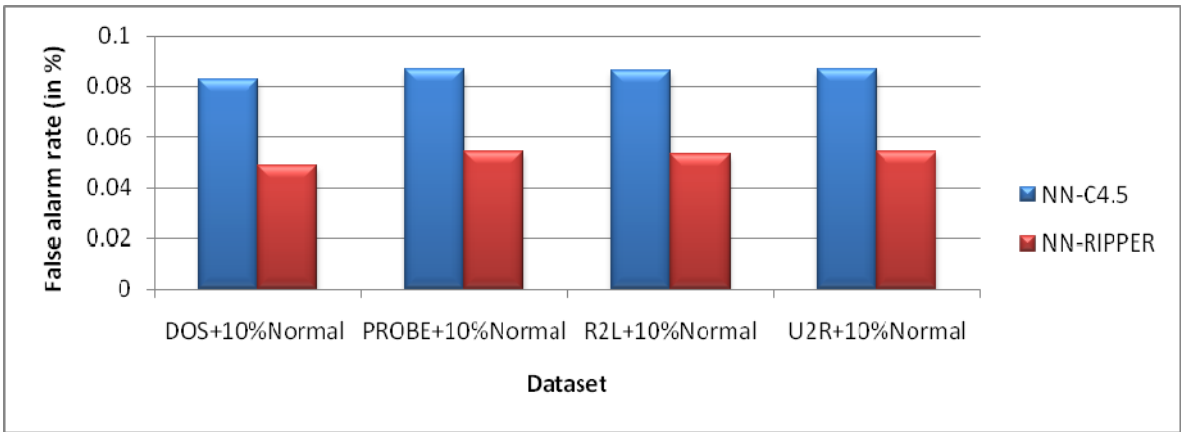

Fig. 3 False alarm rate comparison

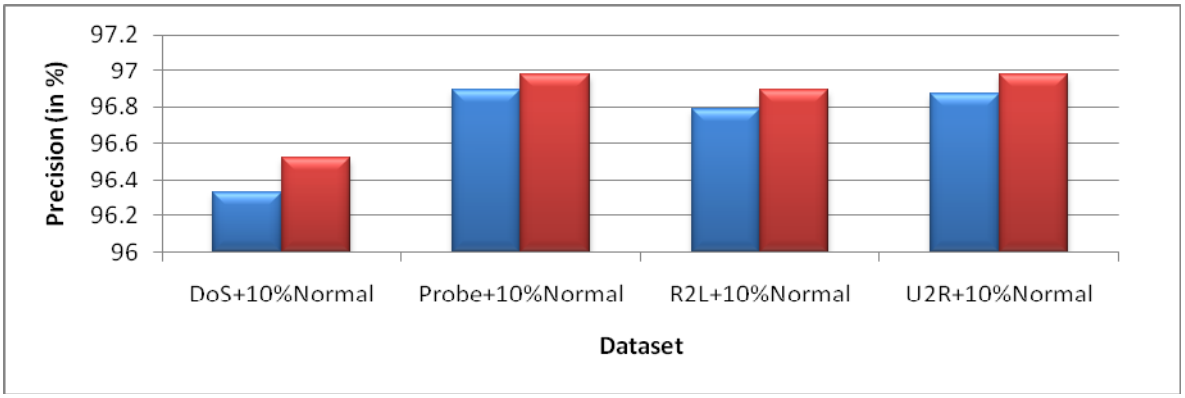

Fig. 4 Precision comparison

VI. CONCLUSION

In this paper, we have presented a metaheuristic bio-inspired approach to select the best features for detecting intrusions. We have studied the impact of features selected using BAT algorithm. We have also analyzed the performance of a Neural Network-RIPPER ensemble classifier on KDDCup'99 dataset and compared it with Neural Network-Decision tree classifier. RIPPER utilized the trained rules from neural network and finally it learns a rule for a given class. It is used to maximize the information gain and number of rules to cover the non- negative rates. The results show that the Neural network with RIPPER achieves better classification accuracy, highest detection rate and lowest false alarm rate of the Intrusion Detection Systems.

REFERENCES

[1] Parpinelli R.S, Lopes H.,. New inspirations in swarm intelligence: a survey. International Journal of Bio-Inspired Computations, 3(1), pp.1-16, 2011.

[2] Kohavi R, John G.H., Wrappers for feature subset selection, Artificial Intelligence, l(2),pp. 273-324, 1997.

[3] Mukkamala S, Janoski G, Sung A, Intrusion Detection: Support Vector Machines and Neural Networks. IEEE International Joint Conference on Neural Networks, St. Louis. MO. IEEE Computer Society Press, pp. 1702-1707, 2002.

[4] Ivor W. Tsang, James T. Kwok, Pak-Ming Cheung, Core Vector Machines: Fast SVM Training on Very Large Data Sets. Journal of Machine Learning Research,. 6,pp.363–392, 2005.

[5] Chebrolu S, Abraham A, Thomas J. P., Feature Deduction and Ensemble Design of Intrusion Detection Systems. Computers and Security, 24, pp.295–307, 2005.

[6] Chen, Y. Li, X-Q. Cheng and L. Guo. Survey and Taxonomy of Feature Selection Algorithms in Intrusion Detection System, Information Security and Cryptology, pp. 153–167, 2006.

[7] Fries T.P, A fuzzy-genetic approach to network intrusion detection, ECCO conference companion on Genetic and evolutionary computation, ACM. New York, pp.2141–2146, 2008.

[8] Makkithaya K, Reddy N.V.S, Acharya U.D, Improved C-Fuzzy Decision Tree for Intrusion Detection, World Academy of Science, Engineering and Technology, 42, pp. 273–277, 2008.

[9] Yang X.-S, A New Metaheuristic Bat-Inspired Algorithm, in: Nature Inspired Cooperative Strategies for Optimization (NISCO 2010) Studies in Computational Intelligence, Springer Berlin ,pp.65-74, 2010.

[10] Yang X.-S, Bat algorithm for multi-objective optimisation. International Journal of Bio-Inspired Computation, 3(5) pp. 267–274, 2011.

[11] R. Y. M. Nakamura, L. A. M. Pereira, K. A. Costa, D. Rodrigues, J. P. Papa X.-S. Yang,"BBA: A Binary Bat Algorithm for Feature Selection" Neural Comput & Application, DOI 10.1007/s00521-012-1028-9

[12] Amir Hossein Gandomi, Xin-She Yang,, "Bat algorithm for constrained optimization tasks", Springer-Verlag, 2012, DOI: 10.1007/s00521012-1028-9.

[13] Koffka Khan, Ashok Sahai,, "A Comparison of BA, GA, PSO, BP and LM for Training Feed forward Neural Networks in e-Learning Context", *Intl. Journal of Intelligent Systems and Applications*,7, pp.23-29, 2012.

[14] Mathew Miller, "Learning Cost-Sensitve Classification Rules for Network Intrusion Detection using RIPPER", *Advanced Intelligent Systems*, Springer, 1999.

[15] Cohen, W., " Fast effective rule induction", *In Proc. of the 12th International Conference on Machine Learning, ICML*, pp.115– 123. Morgan Kaufmann, 1995.

[16] Fürnkranz, J. and Widmer, G., "Incremental reduced error pruning", *In Proceedings the 11th International Conference on Machine Learning, ICML*, pp. 70–77, 1994.

[17] Komviriyavu Makkithaya K., N.V.S. Reddy and U.D. Acharya, "Improved C-Fuzzy Decision Tree for Intrusion Detection", *In Proc. World Academy of Science, Engineering and Technology*, vol. 32, pp. 279–283, 2008.

[18] Ian H Witten, Eibe Frank., Data Mining: Practical machine learning tools and techniques. Second Edition, Morgan Kaufmann Publication, 2005.

[19] J. Han and M. Kamber, "Data Mining: Concepts and Techniques," Morgan Kaufmann, 2006.

[20] Tan Pang-Ning, Michael Steinbach and Vipin Kumar, Introduction to Data Mining. Pearson Education, 2006.

[21] KDDCup 1999 data. http://kdd.ics.uci.edu/ Databases/kddcup99/10 percent.gz.

[22] P Amudha, H Abdul Rauf, "Performance Analysis of Data Mining Approaches in Intrusion Detection", *In Proc. of IEEE Conference on Process Automation Control and computing*, pp.9-16, 2011.