One-Rule Genetic-Fuzzy Classifier

Susan M. Al Naqshbandi
Tilburg centre for Cognition and Communication, Tilburg University
Tilburg, Netherlands
dr.susan_alnaqshbandi@yahoo.com

Venus W. Samawi
College of IT, dept Computer Science
Al al-Bayt University
Mafraq - Jordan
dr_venus2004@yahoo.com

Abstract— **An Intrusion Detection System (IDS) is one of the widely used tools for defending computer networks. Its main goal is to classify activities into two major categories: (1) normal activities and (2) intrusive activities. Both types of activities are hard to predict as the boundaries cannot be well defined and a prediction process may generate false alarms. Many anomaly-based intrusion detection systems have experienced this. However, with *fuzzy logic*, the false alarm rate in determining intrusive activities can be reduced. This paper proposes a *One-rule Genetic-Fuzzy classifier system* to generate the fuzzy rules that are capable of detecting intrusive activities by using Genetic Algorithms (GA). GA is now a viable alternative for the detection of malicious intrusions. They tune the fuzzy membership functions and select an appropriate set of features. After that they generate a proper discrimination rule. Typically, a set of fuzzy rules (fuzzy classifiers) is used to define the normal and abnormal behavior in a computer network. The main goal of this work is to (1) *evolve comprehensible rule(s) that improves the classification rate*, (2) *produce shorter rules*, and (3) *perform automatic feature selection according to the complexity of data*. The proposed system combines both *anomaly-based intrusion detection* and *misuse detection.* A series of experimental results on the well-known KDD Cup 1999 data set [12] demonstrate that the proposed method is feasible. In the paper a performance of the evolved fuzzy classifiers with a classification accuracy of 92% is presented.**

Keywords-Intrusion detection; Genetic algorithms; Fuzzy Logic

I. INTRODUCTION

An intrusion can be defined as ‘‘an act of a person or proxy attempting to break into or misuse a system in violation of an established policy’’ [1]. So, to protect systems from intruders, an Intrusion Detection System (IDS) is needed. It is a combination of software and hardware for monitoring and detecting (1) data traffic and (2) user behavior. IDS is used to identify attempts of illegitimately accessing or manipulating a system which can be done by malware and/or human intruders (crackers, or disgruntled employees).

The main problem to solve is the difficulty of distinguishing between normal behavior and abnormal behavior in computer networks. Here, the big challenge is the significant overlap in data from both types. When trying to detect an Intrusion in a system based on the Anomaly Intrusion Detection, frequently a False Alarm is generated. How to solve this problem?

The past few years have witnessed a rapid growth in the number and variety of applications of fuzzy logic. Fuzzy logic, as a robust soft computing method, has demonstrated its ability in IDS [2-7]. The use of fuzzy logic might reduce the amount of false alarms. Here, it is important to study the features of the process for separation of the overlap between normal and abnormal activities. Fuzzy systems have several important features which make them suitable for IDS [4]. Most fuzzy systems make use of human expert knowledge to create their own fuzzy rule base. Hence, the rule base lacks adaptation. Therefore, building fuzzy systems with learning and adaptation capabilities has received much attention since 2007 [7].

Various methods have been suggested for automatic generation and adjustment of fuzzy rules without using the aid of human experts; the Genetic-Fuzzy System (GFS) is one of the most successful approaches in this regard [7-9].

A GFS is basically a fuzzy system augmented by a learning process based on a Genetic Algorithm (GA). Nowadays, we may consider it a viable alternative for the detection of malicious intrusions [10-11].

Our system is based on this idea. It is implemented as follows. The GA was run over a subset of the data, called the training data, and then tested using one rule over the entire data set to test the real-world performance. The task was to predict the value of each activity (normal or intrusive) for each element out of the dataset containing 311,029 activities.

The rest of this paper is arranged as follows. Sections 2 and 3 illustrate the used dataset and the preprocessing data stage. The fuzzy logic used is pointed out in Section 4. The developed GA including the used coding scheme, fitness function, crossover, and mutation are discussed in Section 5. Section 6 illustrates the One-Rule Genetic-Fuzzy classifier system. Evaluation and testing of the developed software protection system is presented in Section 7. The conclusion of this paper will be found in Section 8.

II. DATA SETS

To evaluate the performance of the proposed IDS, a series of experiments on KDD CUP 1999 ((Knowledge Discovery in Database) dataset was conducted. This dataset is a version of the original 1998 DARPA intrusion detection evaluation program, which is prepared and managed by the Massachusetts Institute of Technology’s (MIT) Lincoln Laboratory [12].An environment was set up to acquire raw TCP/IP dump data for a network by simulating a typical US air force LAN. The LAN was operated like a real environment, but being blasted with multiple attacks. For each TCP/IP connection, 41 various quantitative and qualitative features were extracted. The dataset contains 311,029 activity records.

978-1-4673-0089-6/12/$26.00 ©2012 IEEE

III. PREPROCESING DATASET

The original data set used in the work includes symbolic attributes and numerical attributes. At the beginning the symbolic attributes must be converted to a numerical form. So, all the data become numerical.

Two preprocessing steps have been applied to the original data set used in the work: ***Uniform distribution*** by pattern class and ***normalization of numerical attributes***.

The Uniform distribution process divides the dataset into 10 sets, each set contains 31,102 records generated randomly from the original data set with the following property: **If the sample number of pattern class *k* is *m* and the original data set has *n* samples, then the probability to find a sample of class *k* in the first *n/m* samples of the final data set is 1.0**. Therefore, each portion of the final data set has almost the same distribution of the full data set.

The selected dataset is normalized: **Each numerical attribute in the data set is normalized between 0.0 and 1.0 according to (1):**

$$y = \frac{x - Min}{Max - Min} \tag{1}$$

In which, x is the actual numerical attribute value, y is the numerical value at which $0 \le y \le 1$, *Min* is the minimum value for the attribute that x belongs to, and *Max* is the maximum value for the attribute that x belongs to [2].

IV. FUZZY LOGIC

Fuzzy Logic is a form of many-valued logic; it deals with reasoning that is approximate rather than fixed and exact. In contrast with traditional logic theory (crisp), where binary sets have two-valued logic: true or false, fuzzy logic variables may have a truth value that ranges in degree between 0 and 1. Fuzzy logic has been extended to handle the concept of partial truth, where the truth value may range between completely true and completely false.

In fuzzy logic, fuzzy sets define the linguistic notions, and membership functions define the truth-value of such linguistic expressions [13].

The membership degree to a fuzzy set of an object defines a function where the universe of discourse (set of values that the object can take) is domain, and the interval [0, 1] is the range. That function is called the membership function. "Fig.1" shows the used membership function, i.e., the triangular membership function. A collection of fuzzy sets, called fuzzy space, defines the fuzzy linguistic values or fuzzy classes that an object can belong to.

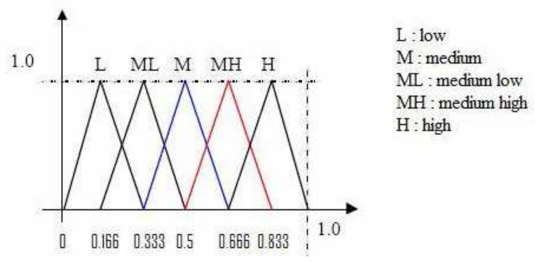


Figure 1: Membership function & Fuzzy space for numerical attributes in the KDD-cup99 data set

With fuzzy spaces, fuzzy logic allows an object to belong to different classes at the same time. The fuzzy space used for each ***numerical attribute*** is assigned as illustrated in "Fig.1". For ***non-numerical*** attributes the categorical values are used as crisp sets (fuzzy sets that does not overlapping each other). Therefore, a value of false for this attribute has a degree of membership to the crisp set FALSE equal to 1.0 and degree of membership to the fuzzy set TRUE equal to zero.

Fuzzy rules have the form:
IF condition *THEN* consequent
Where,

- *Condition* is a complex fuzzy expression, i.e., it uses fuzzy logic operators and atomic fuzzy expressions
- *Consequent* is an atomic expression.

V. GENETIC ALGORITHM

A GA is the computational equivalent of the natural evolutionary process. In a GA, a set of chromosomes (population) is generated, each chromosome codifying a possible solution for the given problem, is evolved using a set of genetic operators (*mutation*, *crossover*, *selection*). Each chromosome has a probability to be used by one or more of the genetic operators, and this probability depends on the adaptability of the chromosome (efficiency of the organism to solve the given problem).

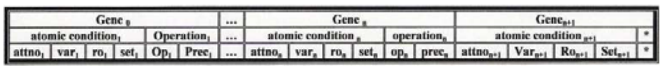
A. Chromosome Encoding (Rule Representation)

The encoding schema used in this paper is ***Direct value encoding***. In ***value encoding***, every chromosome is a string of some values. Values can be anything connected to problem, integer numbers, real numbers or characters to some complicated objects. Since there are different GA for each class, the action part does not need to be represented as part of the rule in the chromosome. It only represents the condition part. Formally, a condition is generated by the following grammars:

(1) <condition> ::= < atomic_cond > <operator><condition>|< atomic_cond ><operator>< atomic_cond >
(2) <atomic_cond> ::= <variable> <rel op> <set>
(3) <operator> ::= $\wedge$<prec> | $\vee$ <prec>
(4) <variable> ::= x_1|…|x_n
(5) <rel op> ::= $\in$ | $\notin$
(6) <set> ::= L | ML | M | MH | H
(7) <prec> ::= 0 | 1 | 2 | 3

In this work, precedence values for each operator in the representation itself (represented by <prec> in the grammar) is introduced. This precedence value indicates the order of evaluation. An operator with a higher precedence value is evaluated first.

The chromosome is defined as a set of n genes. An atomic condition and a fuzzy operator compose a gene, as is shown in "Fig.2". However, there is an exception in the last gene, which is composed of an atomic condition only, while the last part (Fuzzy Operator) is ignored.



Gene $_0$						...	Gene $_n$						Gene$_{n+1}$				
atomic condition$_1$				Operation$_1$		...	atomic condition $_n$				operation$_n$		atomic condition $_{n+1}$				*
attno$_1$	var$_1$	ro$_1$	set$_1$	Op$_1$	Prec$_1$	...	attno$_n$	var$_n$	ro$_n$	set$_n$	op$_n$	prec$_n$	attno$_{n+1}$	Var$_{n+1}$	Ro$_{n+1}$	Set$_{n+1}$	*

Figure 2. Chromosome representation of the condition for n (0...39) gene

In this work, each chromosome is a structure record of length ***>2 and <=41*** Gene (*q*-array), each containing the following fields:

- atomic condition part:
  - attribute number : the attribute number which may be between (0..40) created randomly;
  - var : the attribute value;
  - ro: relational operation which may be ($\in$,$\notin$) created randomly;
  - set: fuzzy set (created randomly) which may be:
    For the numerical fields one of the following sets:
    1. low=[0.0,0.333]
    2. Medium Low=[0.166,0.5]
    3. Medium=[0.333,0.666]
    4. Medium High=[0.5,0.8]
    5. High=[0.666,1.0]

    and for non numerical fields the sets will be as follows:
    1. cr1=[0.0,0.5]
    2. cr2=(0.5,1.0]
- operator part:
  - op: logical operation which may be ($\wedge,\vee$) created randomly;
  - prec: is the number between (0..3) created randomly, it specifies the precedence of evaluating the logical operation between the genes.

An important characteristic of this representation is that, in order to express the genotype, the *operator precedence parser* [14] algorithm is used. This technique allows the evaluation of an expression in a very efficient way. The chromosome only has to be traversed once, that is, the time complexity of the evaluation is *O(n)*, where *n* is the condition expression length.

B. Population Generation

A GA starts with a population of randomly generated chromosomes, and advances towards better chromosomes by applying genetic operators modeled on the genetic processes occurring in nature.

In this step a set of n chromosomes (rule) is randomly generated, consisting of *n* genes, each of which contains fields described earlier.

C. Fitness Evaluation

The GA processes populations of chromosomes, successively replacing one such population with another depending on the fitness function. A fitness function assigns a score to each chromosome in the current population. The fitness of a chromosome depends on how well that chromosome solves the problem at hand.

Before defining the fitness function, it is necessary to see table 1 to recall the concepts on classification rule evaluation concerning ***normal*** and ***attack*** activities [15].

TABLE 1.CLASSIFICATION RULE EVALUATION

Result types	Prediction	actual
True Positive (TP)	normal	normal
False Positive (FP)	attack	normal
True Negative (TN)	attack	attack
False Negative (FN)	normal	attack

The fitness of each chromosome (rule) is evaluated with respect to a set of attribute vectors (training set) to which a class has been previously assigned. In each run of the GA, a rule with different class C_i is evolved. The fitness of a chromosome for the normal class is evaluated according to (2) below [2]:

$$\begin{aligned}
&TP = \sum_{i=1}^{P} predicted\ (normal\_data_i)\\
&TN = \sum_{i=1}^{P} [1 - predicted\ (abnormal\_data_i)]\\
&FP = \sum_{i=1}^{P} predicted\ (abnormal\_data_i)\\
&FN = \sum_{i=1}^{P} [1 - predicted\ (normal\_data_i)]\\
&sensitivity = \frac{TP}{TP+FN},\ specificity = \frac{TN}{TN+FP}\\
&length = 1 - \frac{chrom\_length}{100},\\
&fitness = w_1 * sensitivity + w_2 * specificity + w_3 * length
\end{aligned} \tag{2}$$

The description of the symbols is as follows.

- *TP*, *TN*, *FP,* and *FN* are the true positive, true negative, false positive, and false negative values for the codified rule, respectively.
- *Predicted:* is the fuzzy value of the condition part of the codified rule
- *p* and *q:* are the number of normal and attack samples in the training data set used by each chromosome, respectively,
- *w1*, *w2*, and *w3:* are the assigned weights for each rule characteristic, respectively; three strategies were tested to define the value of the fitness weights. The ***first strategy*** uses the constant values *w1=0.45*, *w2=0.45*, and *w3=0.1*. The ***second strategy*** assigns proportional random values to the weight in order to give more importance to the specificity and sensitivity terms, i.e., the weight *w1*, *w2*, and *w3* are assigned random values such that *w1 >4*w3* and *w2>4*w3.* The ***third strategy*** assigns random values to each weight. In the second and third strategies, the weights are scaled such that the weights sum to one.
  - *normal_data$_i$:* is the subset of normal training patterns, and,
  - *abnormal_data$_i$:* is the subset of attack training patterns.

The set of equations to calculate the fitness for the abnormal class can be obtained by changing abnormal for normal in previous equations. Recall that each individual is associated with a fuzzy prediction value (i.e., fitness value), the best chromosome in the population (i.e., the chromosome with the fittest value) is chosen and the fuzzy rule is added to the fuzzy classifier.

D. Three Genetic Operators

A genetic operator is a process used in GAs to maintain genetic variation. Genetic variation is a necessity for the process of evolution. Genetic operators used in GAs are analogous to those which occur in the natural world: (1) survival of the fittest, or selection; (2) reproduction (crossover, also called recombination); and (3) mutation.

When creating a new population by crossover and mutation, there is a big chance that the best chromosome will be lost. To prevent this we introduce Elitism [16]. ***Elitism is the name of a method, which first copies the best chromosome*** (or a few best chromosomes) ***to the new population***. The rest is done in classical way. Next to elitism, we rely on *Tournament selection* in which a set ***r*** of chromosomes is chosen and compared, the best one being selected for parenthood [17].

When the crossover operator is applied, a crossover point is chosen. The number of genes in the chromosomes should be between two and the minimum length of the two parent chromosomes. The chromosome consists of set of genes at which each gene is represented as a record that contains a set of fields. Therefore, it is important to take into consideration that the crossover point represents the gene number as whole not any of its fields.

After crossover, the offspring is subject to mutation, with probability *pm*. If an offspring is mutated it undergoes either "hoist" mutation or gene randomization with equal probability. Algorithm randomization involves completely randomizing a randomly selected gene of the mutant. Hoisting involves selecting a feature extraction algorithm from the mutant chromosome, selecting a random node from the algorithm's tree, and replacing it by one of the node's descendents. This is less destructive than the algorithm randomization because at least part of the algorithm is preserved.

VI. One-Rule Genetic-Fuzzy Classifier System

Three types of training approaches were suggested in the training phase: ***General Data Splitting method(GDS)***, ***Data filtering Splitting method(DFS)***, and ***Feature Ranking(FR).***

In GDS the whole data set (311,029) is split into two classes (1) Normal activity records these are (60593) and (2) Attack activity records these are (250436). In DFS the whole data is divided into five classes (Normal data 60593, Probing attack 4166, Dos attack 229853, U2R attack 230, R2L attack 16187). In FR ranks the importance of input features for each of the five classes of patterns in the DARPA .

After training our system with one of the above training approches we obtain a rule with a number of features and a fitness value (see table 2).we repeated the training many times and took the most promissing results (see table 2).

In the testing phase one of the rules generated from the training phase is used to classify the normality or abnormality of the pattern.

The input to the classifier is the rule and the pattern to be classified. The process starts by fuzzifying the pattern according to the rule (see below), then calculating the predicted value (by the operator precedence parser [14]). The result is one value. According to that value and a specific threshold the decision can be made (the pattern is either normal or attack) as declared in "Fig.3".

TABLE 2. THE MOST PROMMISSING RULES

Rule Name	Train method	Rule type		No. of features	Fitness
R1	GDS	Normal		3	0.823
R2	GDS	Attack		12	0.910274
R3	GDS	Normal		3	0.87703
R4	GDS	Attack		5	0.8872
R5	GDS	Normal		3	0.84258
R6	GDS	Attack		5	0.88037
R7	GDS	Normal		4	0.84313
R8	GDS	Attack		7	0.90126
R9	DFS	Normal		3	0.9267
R10	DFS	Attack	Dos	11	0.90731
R11	DFS	Attack	Dos	9	0.90913
R12	DFS	Normal		3	0.83969
R13	DFS	Attack	U2R	3	0.81441
R14	DFS	Attack	U2R	3	0.8144
R15	DFS	Normal		4	0.81721
R16	DFS	Attack	Probe	7	0.76179
R17	DFS	Normal		3	0.75923
R18	DFS	Attack	R2L	13	0.77429
R19	FR	Normal		25	0.59752
R20	FR	Attack	Probe	7	0.64399
R21	FR	Attack	Dos	20	0.6033
R22	FR	Attack	R2L	6	0.60331
R23	FR	Attack	U2R	8	0.70483

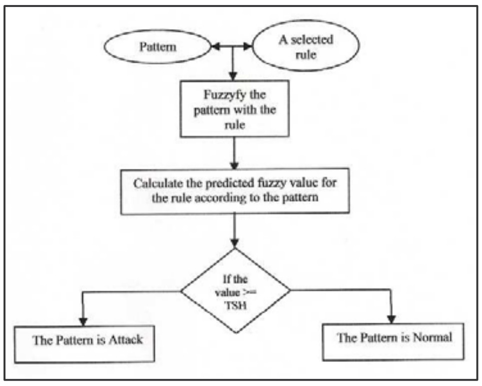

Pattern
A selected rule
Fuzzyfy the pattern with the rule
Calculate the predicted fuzzy value for the rule according to the pattern
If the value >= TSH
The Pattern is Attack
The Pattern is Normal

Figure 3. The flow graph of the One-Rule Classifier

VII. EXPERIMENTAL RESULTS

To study the behavior of the classifier, the whole data set is used for measuring the classification efficiency of the generated rules using (3).

$$C = \frac{S}{T} \tag{3}$$

Where,
C: is the classification efficiency,
S: is the number of correctly classified activities,
T: is the total number of the activity records in the data set.

The main classification idea used in this work depends on the generated rule and the threshold. Therefore it is very important to find out the proper threshold values (a threshold value for normal activities, a threshold value for the attack activities). Choosing the best threshold values can be done by a ***trial-error*** method. Table 3 illustrates that the best threshold value for a normal activity is 1.0 and table 4 illustrates that the best threshold value for the attack activity is 0.01.

TABLE 3. THE RESULTS OF APPLYING THE BEST NORMAL RULE R3 OVER THE WHOLE DATA SET WITH DIFFERENT THRESHOLD VALUES FOR NORMAL ACTIVITIES

Threshold	Correct	Miss	Correctness	Error Rate
1	253658	57371	0.815544531	0.184455469
0.9	245171	65858	0.788257687	0.211742313
0.8	245168	65861	0.788248041	0.211751959
0.7	245027	66002	0.787794707	0.212205293
0.6	245032	65997	0.787810783	0.212189217
0.5	245034	65995	0.787817213	0.212182787
0.4	245038	65991	0.787830074	0.212169926

TABLE 4. THE RESULTS OF APPLYING THE BEST ATTACK RULE R2 OVER THE WHOLE DATA SET WITH DIFFERENT THRESHOLD VALUES FOR ATTACK ACTIVITIES

Threshold	Correct	Miss	Correctness	Error Rate
0.5	271425	39604	0.872667822	0.127332178
0.3	280234	30795	0.90098994	0.09901006
0.2	281796	29233	0.90601198	0.09398802
0.1	284310	26719	0.914094827	0.085905173
0.03	285651	25378	0.918406322	0.081593678
0.01	286876	24153	0.922344862	0.077655138
0.009	286863	24166	0.922303065	0.077696935
0.0085	286864	24165	0.92230628	0.07769372
0.008	286858	24171	0.922286989	0.077713011
0.001	286601	24428	0.9214607	0.0785393
0	250436	60593	0.805185369	0.194814631

The **Correct** field is calculated by adding the *True-positive* to the *True-negative* values for each rule, while the **Miss** field is calculated by adding the *False-positive* to the *False-negative* for each rule. Finally, **Correctness** is calculated using (3), **Error Rate** field equals (1-correctness).

VIII. CONCLUSION

Our main conclusion is that we were able to develop a system that was able to produce adequate detectors which provide a suitable estimation of the amount of deviation from the normal. So, it was possible to apply the classifiers to detect anomalies in real network traffic data. The accuracy of the One-Rule Classifier is 92.00% (see table 4). This is a good result and comparable to those reported in the literature. Moreover, the accuracy can be further improved by applying specific strategies. The success of GA shows that our method of model generation for IDS is a viable alternative. The GA successfully generated an individual model through randomized mutation. The model (generated by training data) was successful in applying its empirical knowledge to data not seen before. This supports the claim that the characteristics of malicious computer activities are inherently dissimilar to normal activities. Despite the fact that only five values for the linguistic variables (see “Fig.1”) were used, the accuracy of the generated classifier rules is trustworthy. During the training process it was found that changing the size of the normal training data set with respect to the size of the attack data set (same percentages of the two data set or different percentages) did not affect the obtained classification rule. Both anomaly detection and misuse detection were supported by this system. It provides the ability to respond to anomalies and not only to signatures of known attacks. The main contribution of the present work is (1) the design of a classification process for the intrusion detection problem; (2) the application of fuzzy logic and genetic algorithms, and (3) its performance of 92%.

ACKNOWLEDGMENT

The authors gratefully acknowledge Professor Dr. Jaap van den Herik for share with us his insight into and expertise on intrusion detection and network security besides his thorough review of the first draft. Moreover, his guidance, teaching and research work have greatly shaped this paper and its follow-on working-process.

REFERENCES

[1] S. Malik,” Network Security Principles and Practices”, chapter 14, November 15, 2002.

[2] J. Gomez, and D. Dasgupta, “Evolving fuzzy classifiers for intrusion detection”, *in: Proceeding of 2002 IEEE Workshop on Information Assurance*, United States Military Academy, West Point, NY, USA, pp. 68–75, 2002.

[3] K. Shah, N. Dave, S. Chavan, S. Mukherjee, A. Abraham, and S. Sanyal, “Adaptive neuro-fuzzy intrusion detection system”, in: Proceeding of A.N. Toosi, M. Kahani / Computer Communications 30 (2007) 2201–2212 2211 IEEE International Conference on Information Technology: Coding and Computing (ITCC’04), 5–7 April 2004, IEEE Computer Society,vol. 1, USA, pp. 70–74, 2004.

[4] J.E. Dickerson, J. Juslin, O. Koukousoula, and J.A. Dickerson, “Fuzzy intrusion detection”, in: Proceeding of IFSA World Congress and 20th North American Fuzzy Information Processing Society (NAFIPS) International Conference. North American Fuzzy Information Processing Society (NAFIPS), 25–28 July 2001, Vancouver, Canada,IEEE Computer Society, vol. 3, pp. 1506–1510, 2001.

[5] M. Gao, and M.C. Zhou, “Fuzzy intrusion detection based on fuzzy reasoning Petri nets”, *in: Proceeding of the IEEE International Conference on Systems*, October 5–8, 2003, *Man and Cybernetics, Washington*, DC, USA, vol. 2, pp. 1272–1277 , 2003.

[6] M.S. Abadeh, J. Habibi, and C. Lucas, “Intrusion detection using a fuzzy genetics-based learning algorithm”, *Journal of Network and Computer Applications*, doi:10.1016/j.jnca,05.002, 2005.

[7] C. Tsang, S. Kwong, and H. Wang, “Genetic-fuzzy rule mining approach and evaluation of feature selection techniques for anomaly intrusion detection”, *Pattern Recognition Society. Published by Elsevier Ltd*. 2007.

[8] M. S. Abadeh, J. Habibi , and E. Soroush, “Induction Of Fuzzy Classification Systems Via Evolutionary Aco-Based Algorithms”, *IJSSST*, Vol. 9, No. 3, ISSN: 1473-804x online, September 2008.

[9] T. Victoire, and M. Sakthivel, “A Refined Differential Evolution Algorithm Based Fuzzy Classifier for Intrusion Detection”, *European Journal of Scientific Research,* ISSN 1450-216X Vol.65 No.2, pp. 246-259, 2011.

[10] D.E. Goldberg, “Genetic Algorithms in Search, Optimization, and Machine Learning”, *Addison-Wesley, Reading,MA*, 1989.

[11] D.E. Goldberg, “The Design of Competent Genetic Algorithms: Steps toward a Computational Theory of Innovation”, *Kluwer Academic Publishers, Dordrecht*, 2002.

[12] KDD-cup data set. http://kdd.ics.uci.edu/data bases /kddcup99/kddcup.html

[13] M. Hellmann, ‘Fuzzy logic introduction”, a Laboratories Antennas Radar Telecom, F.R.E CNRS 2272, Equipe Radar Polarimetrie, France, 2000.

[14] A.V. Aho, R. Sethi, and J.D. Ullman, “Compilers: Principles, Techniques, and Tools”, *Addison-Wesley*, 2001.

[15] M.V. Fidelis, H.S. Lopes & A.A. Freitas," Discovering comprehensible classification rules with a genetic algorithm", Proc. Congress on Evolutionary Computation (CEC), pp. 805-810, 2000.

[16] G. Bakırlı, D. Birant, and A. Kut, “An incremental genetic algorithm for classification and sensitivity analysis of its parameters”,*Expert Systems with Applications*, Volume 38, Issue 3,Pages 2609-2620, March 2011.

[17] C. R.Reeves, and J. E.Rowe, *Genetic Algorithms: Principles and Perspectives A Guide to GA Theory*, Kluwer Academic Publishers, 2003.