IDuFG: Introducing an Intrusion Detection using Hybrid Fuzzy Genetic Approach

Ghazaleh Javadzadeh[1], Reza Azmi[2]
(Corresponding author: Ghazaleh Javadzadeh)

Sharif University of Technology, International Campus, Kish Island, Iran[1]
(javadzadeh@gjdomain.com)
Dept. of Computer Engineering, Alzahra University, Tehran, Iran[2]
(azmi@alzahra.ac.ir)

(Received Apr. 8, 2013; revised and accepted Feb. 23 & June 3, 2014)

Abstract

In this paper, we propose a hybrid approach for designing Intrusion Detection Systems. This approach is based on a Fuzzy Genetic Machine Learning Algorithm to generate fuzzy rules. The rules are able to solve the classification problem in designing an anomaly IDS. The proposed approach supports multiple attack classification. It means that, it is able to detect five classes consist of Denial of Service, Remote to Local, User to Root, Probing and normal classes. We present a two-layer optimization approach based on Pittsburgh style and then combine it with Michigan style. To improve the performance of the proposed system, we take advantages of memetic approach and proposed an enhanced version of the system. We test it on NSL KDD data set to be able to compare our works with previous ones. As results show our approach can converge faster to the classification accuracy about 98.2% and 0.5% false alarm.

Keywords: Computational intelligence, fuzzy genetic, intrusion detection, multiple attack classification, soft computing

1 Introduction

Intrusion Detection System (IDS) is a security tool that detects a malicious behavior called attack. An IDS may be a software or a combination of software and hardware [13]. IDS monitors and analyzes the data within a computer or a network to detect and alert the system if an attack happens. It should be mentioned that an IDS do not react against attacks, but it just detect and alert the system or the network administrator. The function of reacting against attacks is undertaken by other security mechanisms. However, IDS plays a critical role in security of networks and systems since detection is the first step in controlling an attack.

We can classify IDS from different aspects, the most common is the source of audit data; accordingly there are two types of IDSs: Host based (HIDS) and Network based (NIDS). HIDS surveys the data of a single host such as the operating system kernel logs, and application program logs;While NIDS gathers and analyzes the data transmit between several hosts in a computer network. Actually a NIDS examines the traffic packets to detect intrusion patterns. Although, each of these systems has its own advantage and disadvantage. It is recommended to use a combination of both for more security. Figure 1 illustrates the Host based and the Network based intrusion detection system.

There are several detection methods that an IDS use to detect intrusions. Two main detection methods are Signature based (misuse) and Anomaly based detection. In the former, the IDS compares the data traffic with

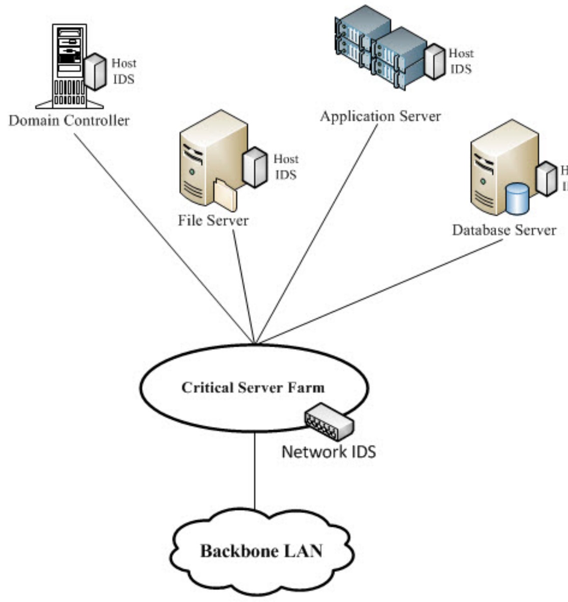


Figure 1: Host based and network based IDS

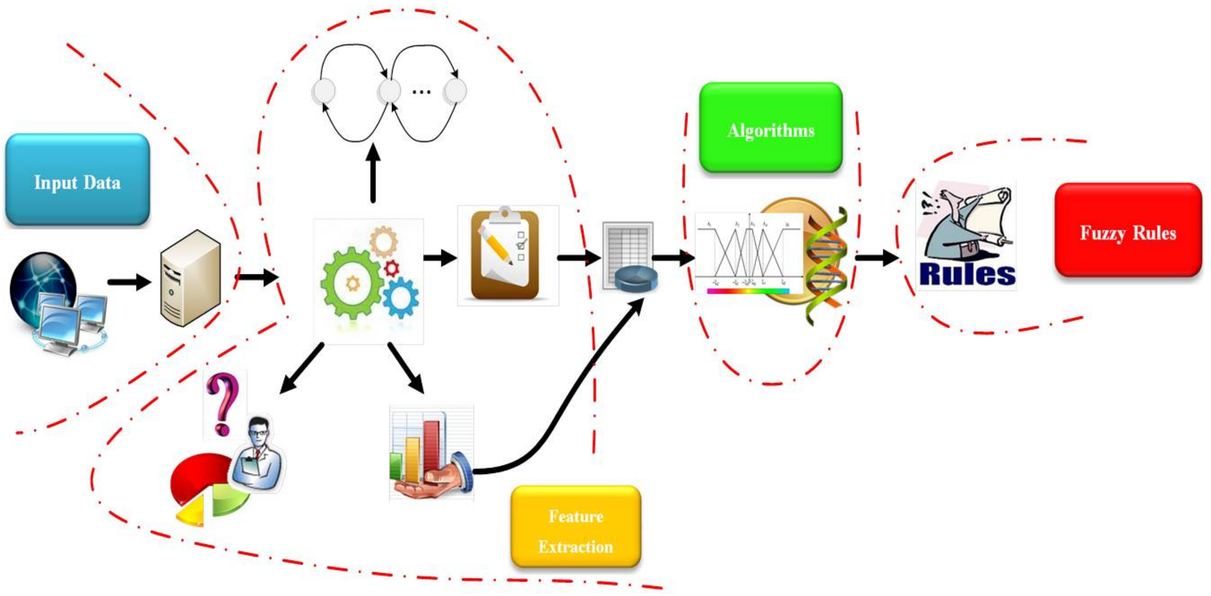


Figure 2: Host based intrusion detection system

some predefined patterns of attacks, and if they match, the traffic will be detected as an attack. In this method the rate of false alerts reduces, however attacks with new patterns cannot be detected.

Anomaly detection method detects data traffic as an attack if it is not matching the normal traffic. In fact, a normal traffic is defined in a system, and IDS captures and compares the traffic with it. If a mismatch happens, it means that an attack happens. In this method more attacks with new patterns can be detected while we may encounter more false alarms too. In this paper we study both types of IDSs but propose an anomaly detection approach for NIDS.

There are two important challenging points when reviewing the IDS literature, the dataset and the algorithm. Selecting a proper algorithm to train the IDS and using suitable dataset to test it, affect the detection ability. Many researchers have done efforts in this domain. Numerous methods for detection are invented with the goal of high classification accuracy, beside low false alarm, high computational speed, and low computational complexity. Figure 2 illustrates a host based intrusion detection system.

The first challenging point is selecting a proper algorithm to detect intrusion. In most of the previous works the detection results in a binary state which shows the traffic is an attack or not. Besides we find out that an action is an attack, it is important to know the type of that attack, because different type of attacks threat different security aspects. Some security attacks, disclose the confidentiality of our assets while some others alter integrity or the impossible availability of them. When we know the attack type, we can pick the most proper security mechanism to encounter with it.

In this paper, we propose an algorithm with different structures for multiple classification of the attack traffics, means we present a multiple classification of detection results. For example, it classifies the traffic as the normal user behavior, and other of the intrusion types such as Probing, Denial of Service (DOS), and User to Roots attacks (U2R), and so on. Actually the type of attack will be determined in this kind of classification. The continual changing nature of attacks requires a flexible defensive system that is capable of analyzing the enormous amount of traffic in a manner which is less structured. The ability of performing multiple attack classifications is one of the most important features of our proposed approach. The difference between multiple classification and binary classification is illustrated in Figure 3.

Studies show that statistical techniques such as Hidden Markov Model, Multivariate Adaptive Regression Splines, Bayesian Classifier and Classification and Regression Tree have been applied to intrusion detection. These statistical approaches usually result in an inflexible detection system that is unable to detect an attack if the sequence of events is different a little bit from the predefined profile. Recently, computational intelligence has been successfully applied for developing IDS.

As mentioned in [25] characteristics of computational intelligence systems, such as adaption, fault tolerance, high computational speed and error resilience in the face of noisy information, fit the requirements of building a good intrusion detection model. According to the classification of their article, in order to design an algorithm for an IDS using computational intelligence we can apply following techniques: artificial neural networks, fuzzy sets, evolutionary computation, artificial immune system, swarm intelligence, and soft computing. Figure 4 represents the mentioned classification of these techniques.

Among these techniques Soft Computing has a particular place. In fact, soft computing is a method in which, combination of other computational intelligence

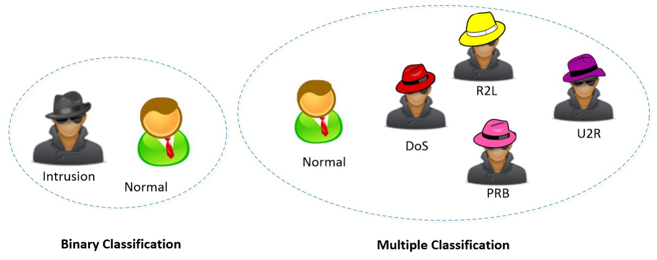


Figure 3: Multiple classifications against binary classification

techniques are used. As a soft computing approach, we have chosen a mixture of evolutionary computation and fuzzy systems for designing an intrusion detection algorithm. There are two major reasons for using fuzzy in IDS. First, there are many numeric attributes in the collected audit data and various derived statistical measures. Second, since fuzzy logic deals with imprecise information and the essence of security includes fuzziness as boundaries between normal and abnormal are not well define.

Genetic Base Machine Learning or GBML is a type of Evolutionary Algorithm (EA); EA is a family of search and optimization approaches that is inspired of the natural principles of evolution [22]. Actually, we proposed an intrusion detection system for known and unknown attacks (anomaly detection) using one of the computational intelligence techniques, as shown in Figure 4.

Among all computational intelligence techniques, we have chosen a mixture of evolutionary computation and fuzzy systems called GBML algorithm [14]. GBML has two main approaches: Pittsburgh and Michigan. Each of them has some advantages and disadvantages which have been mentioned in detail in [22]. Our proposed algorithm is a combination of these two approaches; by this means it would be possible to use the advantage of both algorithms such as the direct optimization of rules, high search ability, and at the same time, shorter computational time. In fuzzy genetic algorithms, each individual can be a rule or rule set depending on the approach. Pittsburgh style handles the rule sets as individuals while rules are considered as individuals in Michigan.

As mentioned above, it is important to use a proper dataset for the test phase. Of course, the dataset contains features of normal and abnormal behavior use for designing and testing intrusion detection systems. There are two way of selecting datasets. The first one is to create your own dataset and the second is using existing datasets such as KDD Cup99 [1] or NSL KDD [2]. Many researchers used KDD CUP99 dataset (KDD). Mahoney and Chani [20] criticized the dataset validity. As they claim information in the dataset does not look like a real traffic in many aspects. Based on their analysis, the IDS created by using KDD with low false alarm may generate high false alarms in real environment.

NSL KDD dataset is an improved version of KDD CUP 99 [2]. This dataset has been used for the evaluation of anomaly based detection methods. The data set was generated by gathering the network logs. It contains two parts, train and test. Patterns of this dataset contain 41 attributes. Figure 5 shows a snapshot of NSL KDD dataset.

The best advantage of using existing datasets is that we can evaluate our work with others. But actually because none of the existing datasets contains all kind of todays attacks it is recommended to create your own dataset with all possible kinds of attacks. Also for different websites and web applications we may encounter different attacks, so it is better to gather the normal and abnormal behaviors of their users for each one separately. In this paper we use the NSL KDD to test our proposed algorithm, because it let us compare our algorithm with previous works.

Briefly, in this paper, we propose an intrusion detection system called IDuFG that is capable of doing multiple attack classifications. The algorithm of this system uses a soft computing technique that is a fuzzy genetic approach. Test and train of the proposed algorithm is been done on NSL KDD dataset.

The rest of this paper is organized as follows: Section 2 is the literature review and also an overview of the related work in terms of intrusion detection systems. In Section 3 and Section 4, the proposed approach and simulation results are explained respectively. Section 5 concludes the paper and presents the future work.

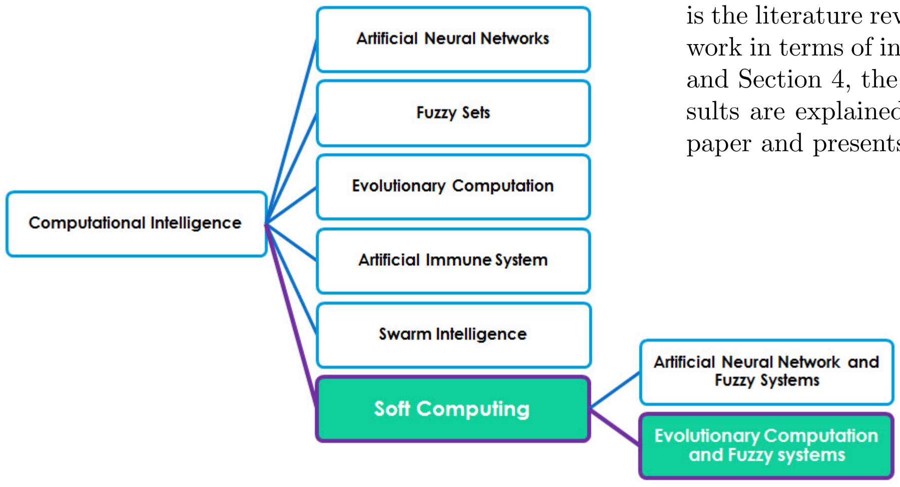
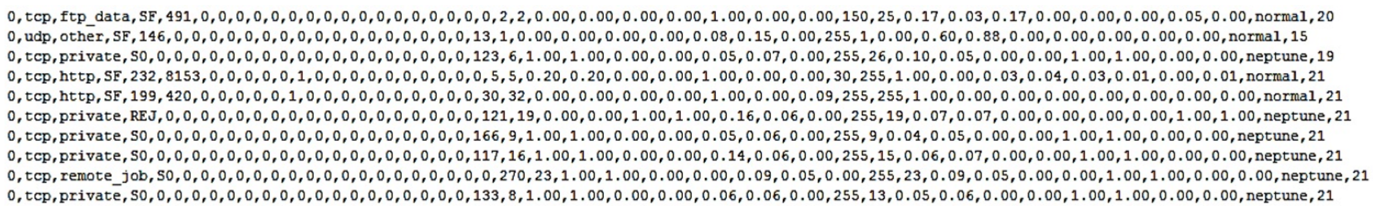

Computational Intelligence
Artificial Neural Networks
Fuzzy Sets
Evolutionary Computation
Artificial Immune System
Swarm Intelligence
Soft Computing
Artificial Neural Network and Fuzzy Systems
Evolutionary Computation and Fuzzy systems

Figure 4: Computational intelligence techniques



```
0,tcp,ftp_data,SF,491,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,0.00,0.00,150,25,0.17,0.03,0.17,0.00,0.00,0.00,0.05,0.00,normal,20
0,udp,other,SF,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,13,1,0.00,0.00,0.00,0.00,0.08,0.15,0.00,255,1,0.00,0.60,0.88,0.00,0.00,0.00,0.00,0.00,normal,15
0,tcp,private,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,123,6,1.00,1.00,0.00,0.00,0.05,0.07,0.00,255,26,0.10,0.05,0.00,0.00,1.00,1.00,0.00,0.00,neptune,19
0,tcp,http,SF,232,8153,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,5,5,0.20,0.20,0.00,0.00,1.00,0.00,0.00,30,255,1.00,0.00,0.03,0.04,0.03,0.01,0.00,0.01,normal,21
0,tcp,http,SF,199,420,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,30,32,0.00,0.00,0.00,0.00,1.00,0.00,0.09,255,255,1.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,normal,21
0,tcp,private,REJ,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,121,19,0.00,0.00,1.00,1.00,0.16,0.06,0.00,255,19,0.07,0.07,0.00,0.00,0.00,0.00,1.00,1.00,neptune,21
0,tcp,private,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,166,9,1.00,1.00,0.00,0.00,0.05,0.06,0.00,255,9,0.04,0.05,0.00,0.00,1.00,1.00,0.00,0.00,neptune,21
0,tcp,private,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,117,16,1.00,1.00,0.00,0.00,0.14,0.06,0.00,255,15,0.06,0.07,0.00,0.00,1.00,1.00,0.00,0.00,neptune,21
0,tcp,remote_job,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,270,23,1.00,1.00,0.00,0.00,0.09,0.05,0.00,255,23,0.09,0.05,0.00,0.00,1.00,1.00,0.00,0.00,neptune,21
0,tcp,private,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,133,8,1.00,1.00,0.00,0.00,0.06,0.06,0.00,255,13,0.05,0.06,0.00,0.00,1.00,1.00,0.00,0.00,neptune,21
```

Figure 5: An NSL KDD dataset snapshot [2]

2 Litrature Review

In our research, we study different approaches that have been used in designing intrusion detection systems. Numerous approaches consist of pure or hybrid, have been developed in recent years. It seems that hybrid algorithms that use a combination of fuzzy logic and other methods are attracting issues in the area of designing intrusion detection systems.

Genetic algorithm includes search algorithms that are inspired from biological evolution as a problem-solving strategy by using some natural mechanism such as generation, mutation, crossover, and selection. Several Genetic Based Machine Learning (GBML) algorithms have been proposed for designing fuzzy rule-based systems. As mentioned in [5] GAs are search algorithms that theoretically and empirically are proven to have a great capability in complex space in search and optimization problems, so will be used in pattern classification problems commonly. Also brilliant characteristics of genetic algorithm, such as automatic optimizing, global researching, and adaptability, is the reason for choosing it.

Several fuzzy GBML algorithms have been proposed for designing fuzzy rule-based systems. Fuzzy GBML algorithms can be also classified into two categories: Michigan approach and Pittsburgh approach. In [22] Nojima et al. developed two GA-based schemes for the design of fuzzy rule-based classification systems. Search ability of fuzzy genetic rule selection and fuzzy GBML algorithms compare to each other.

Cintra and Camargo in [8] named Rule Based definition as one of the most important and difficult tasks in designing fuzzy systems. They introduce a fuzzy rule base generation method using the genetic algorithm. The algorithm support binary classification. This algorithm includes a phase of pre-selection of candidate rules that has been proposed by the authors. The use of a self-adaptive algorithm for the fitness calculation in genetic algorithm is proposed as an improvement of the mentioned method. The advantage of proposed method has been tested by some experimental results.

A parallel genetic local search algorithm is presented in [4]. According to that paper, an important solution to reduce the false alarm rate in detection intrusions is the fuzzy logic. This algorithm produces fuzzy rules for network intrusion detection systems. This system use Michigan approach. As shown in Figure 6, in this algorithm, the total population is divided into subpopulations. Each subpopulation contains the same class fuzzy rules.

An intrusion detection approach that extracts accurate and interpretable fuzzy rules for classification has been proposed in [23]. They use a statistical pattern recognition approach to design an intrusion detection system. These rules are obtained from the network traffic data. They evaluated their approach using KDD CUP and compared the results with some well-known classifiers. In that paper fuzzy rule generation strategy is discussed. The problem of this method is high false alarm ratio.

A genetic fuzzy system is proposed in [6] and the authors compared it with other approaches in intrusion detection systems. As the results show, it is obvious that because false alarm rate in a fuzzy genetic system, the based intrusion detection is lower than other approaches, IDS which develops using genetic fuzzy systems would be more reliable than other approaches.

Also, a new genetic fuzzy logic method for automatic rule generation has been proposed in [26]. This algorithm automatically adjusts the crossover rate and the mutation rate using rules population diversity and evolutionary speed. The simulation results in that paper indicate that the algorithm is practical and effective. The disadvantage of this algorithm is that in cases that the boundaries of samples are not clearly defined, the rules of different

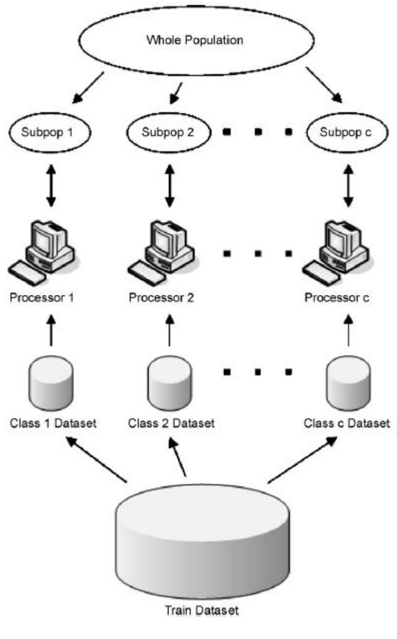


Figure 6: The parallel learning framework [4]

classes overlap with each other.

A hybrid algorithm proposed by H. Ishibuchi in [14] that has the advantages of the two fuzzy GBML algorithms. The basis of the algorithm is Pittsburgh that Michigan-style algorithm is applied to each rule set after the mutation operation to generate new fuzzy rules. Another hybrid algorithm was proposed in [15] for designing fuzzy rule based systems for pattern classification problem. This algorithm is a combination of two Genetic-Based Machine Learning algorithms called Pittsburgh and Michigan. The experimental results show that this algorithm has high search ability that Michigan and Pittsburgh approaches. To obtain a better tradeoff between the accuracy of the system and its complexity, this algorithm should be extended to the multi objective design of fuzzy rule-based classification system. The algorithms have high complexity.

In this paper, since most of the related work and similar algorithms use KDD dataset to test their proposed methods, we also test our proposed algorithms on NSL KDD to be able to compare our work with them. As our studies shows, each of the previous works has its own advantages and disadvantages. None of them covers all of the requirements of a good IDS. As shown in Table 1, the most important challenging issues in designing an IDS are performance, detection rate, computational time, ability to detect new attacks, and false alarm.

Table 1: Comparison of fuzzy genetic systems

	Method	Category	Pros	Cons
[8]	Fuzzy Rules Generation using Genetic algorithms with Self-adaptive Selection	Fuzzy Genetic	Speed up the search process	False alarms
[4]	Parallel Genetic Local Search	Fuzzy Genetic	High performance	Long computational times
[23]	Genetic Fuzzy Rule Mining Approach	Fuzzy Genetic	High detection rate	Long computational time
[6]	Genetic Fuzzy System Based	Fuzzy Genetic	High performance	Unable to detect new attacks
[26]	Automatic fuzzy rule generation using fuzzy genetic algorithm	Fuzzy Genetic	Practical and effective in applications	Long computational time
[15]	Hybridization of Fuzzy GBML approaches	Fuzzy Genetic	High search ability	There is no tradeoff between accuracy and complexity of system

3 Proposed Approaches

We propose a fuzzy based genetic algorithm for designing Intrusion Detection Systems in two Basic and Enhanced approaches. Our basic proposed approach is a combination of two Genetic Based Machine Learning styles called Michigan and Pittsburgh; by this means it would be possible to use the advantage of both styles such as direct optimization of rules, high search ability, and at the same time shorter computational time. In this section, we briefly introduce genetic algorithm, Pittsburgh and Michigan styles, fuzzy rules, and finally our proposed fuzzy-based approaches will be described in the last section.

3.1 Genetic Algorithm

Here, we describe the general process of the genetic algorithm. Genetic algorithms are an effective search method in cases that the solution space is wide and large that lead to find optimized solutions of a problem. This algorithm works with a series of coded variables. So we should code our variable to be able to find the optimized solution for them. In using the genetic algorithms, three following concepts should be determined:

1) Defining the objective function or the cost function
2) Genetic representation
3) Defining and implementation of the genetic operators

The general process of the genetic algorithm is illustrated in Figure 7; in this algorithm, at first an initial population is generated randomly, then the fitness of each individual is evaluated. According to the fitness value of each one two parents are selected. Then offspring will be produced by using the genetic operator such as crossover and mutation. This offspring is an individual of the next generation.

In subsequent iteration the initial population is an improved version rather random. At the end of iterations stopping condition is checked. As studies show there are several stopping criteria such as:

1) A fixed number of iterations
2) A fixed amount of time
3) Run until convergence has occurred

Genetic algorithm is applied on a set of solutions called population. Usually populations consist of 20 to 100 individuals. Most custom method to indicate individuals in genetic algorithm is in a string form. One of the important points that should be considered is that after applying ge-

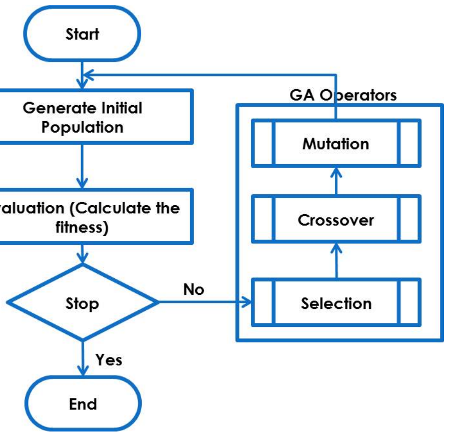

Start
Generate Initial Population
Evaluation (Calculate the fitness)
Stop
No
Yes
End
GA Operators
Mutation
Crossover
Selection

Figure 7: Fuzzy genetic algorithm flowchart

netic operators and producing new results, it is possible to reach the answers that are not satisfying the problem conditions. A simple solution to this problem is using penalty function, by this means fitness of the individuals that not match the conditions will decrease excessively.

Usually, to produce next generations, three fundamental operators are used: Selection, Crossover, and Mutation. The new generation is assessed according to the stopping condition. If the stopping condition is not satisfied this process will repeat to generate a more optimal generation. There are different kinds of operators; according to our studies and their comparisons between several genetic operators, we use tournament selection and uniform crossover, in our proposed approach.

3.2 Fuzzy GBML Styles

Fuzzy GBML algorithms have two main categories: Michigan and Pittsburgh. According to [15], in Michigan-style algorithms, there is a population consisting of a pre-specified number of fuzzy rules. A population of individuals generates new populations by the operators such as crossover and mutation. The procedure of Pittsburgh algorithm is the same as Michigan with partial differences.

Each individual in Michigan is a rule and the population is a collection of these rules. As a comparison of these two styles: in Pittsburgh individual is a rule set, fitness define for a rule set, optimization is direct, computation time is long and memory usage is large while in Michigan, the individual is a rule, fitness define for a rule. Also it has indirect optimization, with short computation time and small memory usage [6]. The differences of these two approaches are summarized in Table 2.

Michigan.
The Michigan approach is characterized by the fuzzy rules as its individuals, and the whole population is the solution to the classification problem [12]. Detail description of coding rule, operators and fitness calculation is available in [9]. Here we just point to the general form of a

Table 2: Comparison of Pittsburgh and Michigan [16]

Approach	Pittsburgh	Michigan
Individual	A rule set	A rule
Fitness definition	For a rule set	For a rule
Optimization	Direct	Indirect
Elite	Rule sets	Rules
Inheritance of good rules	×	√
Inheritance of good rule sets	√	×
Computation time	Long	Short
Memory storage	Large	Small

Michigan approach. According to Michigan Style Fuzzy GBML Algorithm mentioned in [15], the approach is as follow (Algorithm 1):

Algorithm 1 Michigan style fuzzy GBML
```
1: Begin
2: Generate $N_{rule}$ fuzzy rules as initial population.
3: Calculate the fitness value of each rule.
4: Generate $M$ rules using genetic operations.
5: Use $M$ new rules and $(N_{pop} - M)$ best rules of current
   population to produce the next generation.
6: If the stopping condition is not satisfied go to step 2.
7: End
```

As it is obvious, in all steps of this algorithm, fuzzy rules are considered as individuals. Therefore the fitness value is calculated for each rule and also the operators apply on rules.

Pittsburgh.
In this approach each individual is a complete solution to the problem. A complete solution means a rule set. Detail description of coding rule, operators and fitness calculation is available in [9]. Here we just point to the general form of a Pittsburgh approach. According to Pittsburgh Style Fuzzy GBML Algorithm mentioned in [15], the approach is as follow (Algorithm 2).

Algorithm 2 Pittsburgh style fuzzy GBML
```
1: Begin
2: Generate $N_{pop}$ rule sets with $N_{rule}$ fuzzy rules as ini-
   tial population.
3: Calculate the fitness value of each rule set.
4: Generate $M$ rule sets using genetic operations.
5: Use $M$ new rule sets and $(N_{pop} - M)$ best rule sets of
   current population to produce the next generation.
6: If the stopping condition is not satisfied go to step 2.
7: End
```

To takes advantage of both approaches; we use a combination of them in our proposed approaches. Michigan yields good rules but not necessarily good rule-sets while Pittsburgh yields good rule sets but not good rules. But our proposed algorithm enjoys good rule sets with good rules. Also this algorithm has the high search ability of the Michigan, as well as the direct optimization ability of the Pittsburgh.

3.3 Fuzzy Rules

In fuzzy genetic algorithms, each individual can be a rule or rule set depending on the approach. Pittsburgh style handle the rule sets as an individual while rules are considered as individual in Michigan. In this section, the procedure of rule generation is described. This process is one of the important steps in our proposed algorithm. In our proposed algorithms we use fuzzy rules of the following type:

$$R_i = if\ X_1\ is\ A_{i1}\ \&\ ...\ \&\ X_n\ is\ A_{in}$$
$$then\ class\ C_i\ with\ CF_i\ ,\ i = 1, 2, ..., n. \quad (1)$$

Where R_i is a fuzzy rule and X_i refers to the i^{th} feature of the corresponding pattern, A_{in} is an antecedent fuzzy set with linguistic label, C_i is a consequent class and CF_i is a rule weight and n is the number of rules.

In this stage we started to find the class of each rule [19]. To do so, the compatibility of antecedent fuzzy rules with the random pattern is calculated by equation.

$$\mu_i(X_P) = \mu_i(X_{P1}) \times \ldots \times \mu_{in}(X_{Pn}), P = 1, 2, \ldots, m \quad (2)$$

where $\mu_{in}(.)$ is the membership function of A_{in} that achieve from five linguistic values shown in Figure 8 and m is the number of patterns. The consequence class of each rule is determined according to following equation. Figure 9 shows a sample of a fuzzy rule.

$$\beta_{class\hat{h_i}}(R_i) = max\{\beta_{class_1}(R_i), \ldots, \beta_{classh}(R_i)\} \quad (3)$$

where

$$\beta_{classh}(R_i) = \frac{\sum_{xp \in classh \mu_i(xp)}}{N_{classh}}$$
$$h = 1, 2, \ldots, n(numberofclasses). \quad (4)$$

Each of the fuzzy rules in the final classification has a certainty grade (class weight), that means the strength of that fuzzy rule and calculated according to:

$$CF_j = \frac{(\beta_{class\hat{h_i}}(R_i) - \overline{\beta})}{\sum_{h=1}^{5} \beta_{classh}(R_i)'} \quad (5)$$

where

$$\overline{\beta} = \frac{\sum_{h \neq h_i} \beta_{classh}(R_i}{n-1}. \quad (6)$$

Certainty grade of each fuzzy rule is a number in [0,1] interval that indicates the accuracy amount of the consequence part of a rule according to the accuracy of the antecedent part of that rule. So rules can be generated accordingly and also the rule sets that are a group of rules.

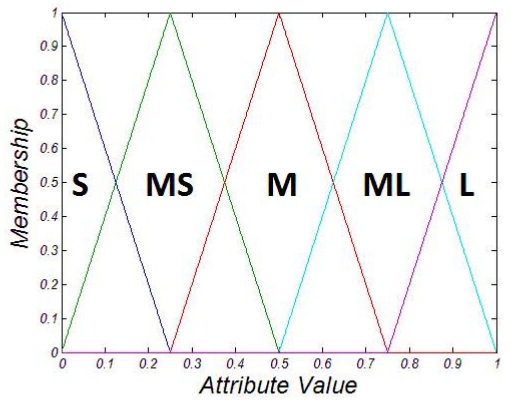


Figure 8: Membership function of five linguistic value [5]

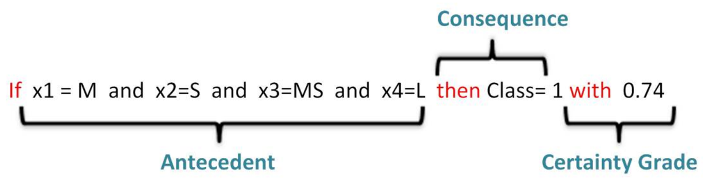

Consequence
If x1 = M and x2=S and x3=MS and x4=L then Class= 1 with 0.74
Antecedent
Certainty Grade

Figure 9: Fuzzy if-then rule

According to the approach that we pick, rules or rule sets are considered as individuals.

After generating N_{pop} rule sets with N_{rule} fuzzy rules as initial population using the above method, we should evaluate the fitness of each individual. Fitness value for each rule is calculated by (N_{pop} and N_{rule} indicate the number of rule sets and the number of rules in a rule set):

$$fitness(R_j) = w_t * T_p - w_f * F_p \quad (7)$$

Where T_p is the number of correctly classified training patterns and F_P is the number of incorrect classified training patterns but the fitness of the rule set is just the number of the correctly classified patterns. w_t and w_f are the weights of correct or incorrect classification, respectively. In fact, each rule is evaluated by classifying the given training patterns.

One of the important points that should be considered is that after applying genetic operators and producing new results, it is possible to reach the answers that are not satisfying the problem conditions. A simple solution to this problem is using penalty function, by this means fitness of the individuals that not match to the conditions will decrease excessively. In the above equation the w_f plays the role of penalty function.

Since any alteration in membership functions leads to vague results, in this paper we do not apply any changes in membership function of input. In the proposed fuzzy GBML algorithm, we use the above membership function for all inputs.

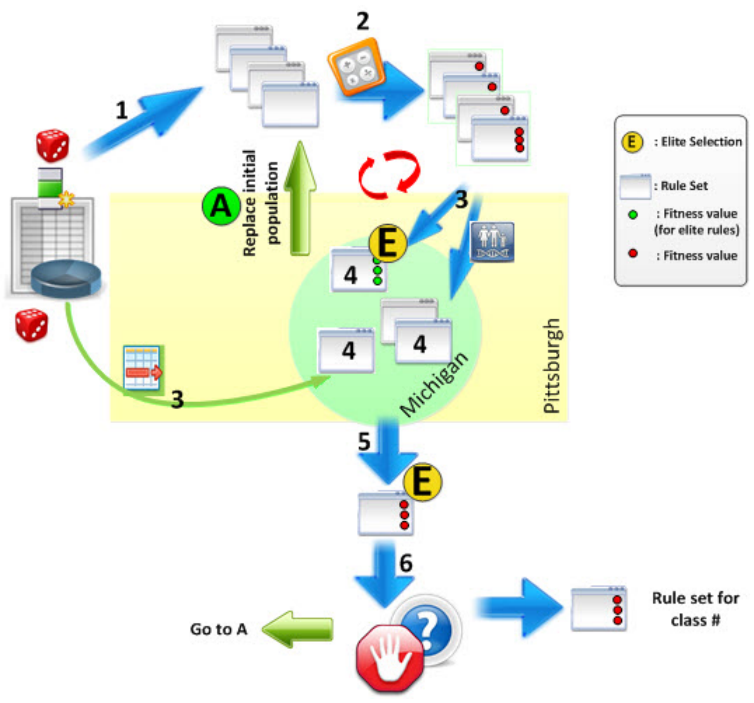


Figure 10: IDuFG algorithm for each class (Numbers 1-6 indicates the algorithm steps)

3.4 The Proposed Approaches

We propose a fuzzy based genetic algorithm for designing Intrusion Detection Systems that is divided in Basic and Enhanced approaches. Our basic proposed algorithm is a combination of these two approaches; by this means it would be possible to use the advantage of both algorithms such as direct optimization of rules, high search ability, and at the same time the shorter computational time. Accordingly in Subsection 3.4.1 we explain the Basic approach and the enhanced approach will be explained in Subsection 3.4.2.

3.4.1 Basic Approach

The goal of a fuzzy GBML algorithm is to find the small number of fuzzy if-then rules with high classification anility. Our proposed algorithm is a combination of two fuzzy GBML algorithms; Pittsburgh is selected as the base of the algorithm and at the end of iterations of Pittsburgh we have a single iteration of Michigan. Initial population is generated according to the random training patterns. N_{pop} rule sets are generated from the random patterns of the training data set as the initial population. Figure 10 illustrates the process of result rule set generation for each class. The algorithm is running for each class separately, and the final result contains all rule sets of all classes. Our proposed algorithm is written in Algorithm 3.

To produce an initial population of fuzzy rule sets, we pick N_{pop} training patterns randomly, and generate combinations of antecedent part of several fuzzy rules according to the selected patterns.

Authors of [7] discussed about how to extract fuzzy rules directly from numerical data for pattern classification. Each rule set is generated according to a training

Algorithm 3 Basic approach

```
1: Begin
2: Generate $N_{pop}$ rule sets with $N_{rule}$ fuzzy rules.
3: Calculate the fitness value of each rule set in the current population.
4: Generate next generation according to the following portions (experiments show that the following percentage producing best results):
       • 80% rule sets of the next generation by the selection, crossover and mutation in the same manner as the Pittsburgh-style algorithm.
       • 10% of the next generation is formed by elite rule set selection.
       • 5% is elite rule selection among all rule sets (i.e. the best rules of all rule sets).
       • 5% is the updated patterns from the training patterns that not choose in the first step.
5: Apply a single iteration of the Michigan style algorithm (i.e., the rule generation and the replacement) to each of the generated rule sets.
6: Choose the best rule set in the new population as the result of this iteration.
7: Return to Step 2 if the pre-specified stopping condition is not satisfied.
8: End
```

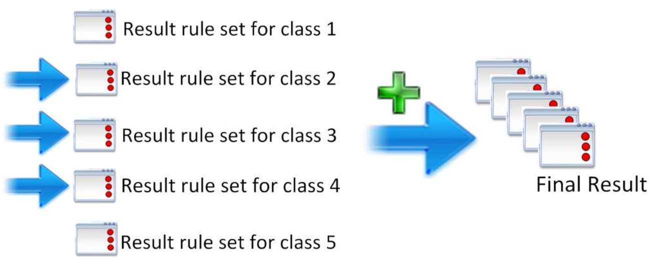

Result rule set for class 1
Result rule set for class 2
Result rule set for class 3
Result rule set for class 4
Result rule set for class 5
Final Result

Figure 11: Final result of basic approach

pattern. As mentioned before, an algorithm is running for each class separately; so in all steps the rules which are in the same class of algorithm running for will be accepted.

In Step 3 that we apply genetic algorithms, we use tournament selection for selecting the parent rules or rule sets, by this means we give the chance of contribution in forming the next generation to the rules or rule sets that have not the high fitness value. In this method a subset of individuals is randomly selected, and then these individuals compete with each other according to their fitness . To achieve the offspring we use uniform crossover with probability of Pc and mutation with probability of Pm. Each part of individuals is mutated independent of other parts. It means the mutation of a rule do not influence on the mutation probability of other parts.

After producing the next generation we should check the stopping condition. If the pre-specified stopping condition is not satisfied we should return to the step 2; in our proposed basic approach, the number of iterations is considered as the stopping conditions. This time instead of starting with a random population, use the newly produced generation. The Result of the algorithm running for each class is the best rule set which means the rule set with the highest fitness value. As shown in Figure 11 the final result of the algorithm contains all result rule sets of all classes. The flowchart of the basic approach is illustrated in Figure 12.

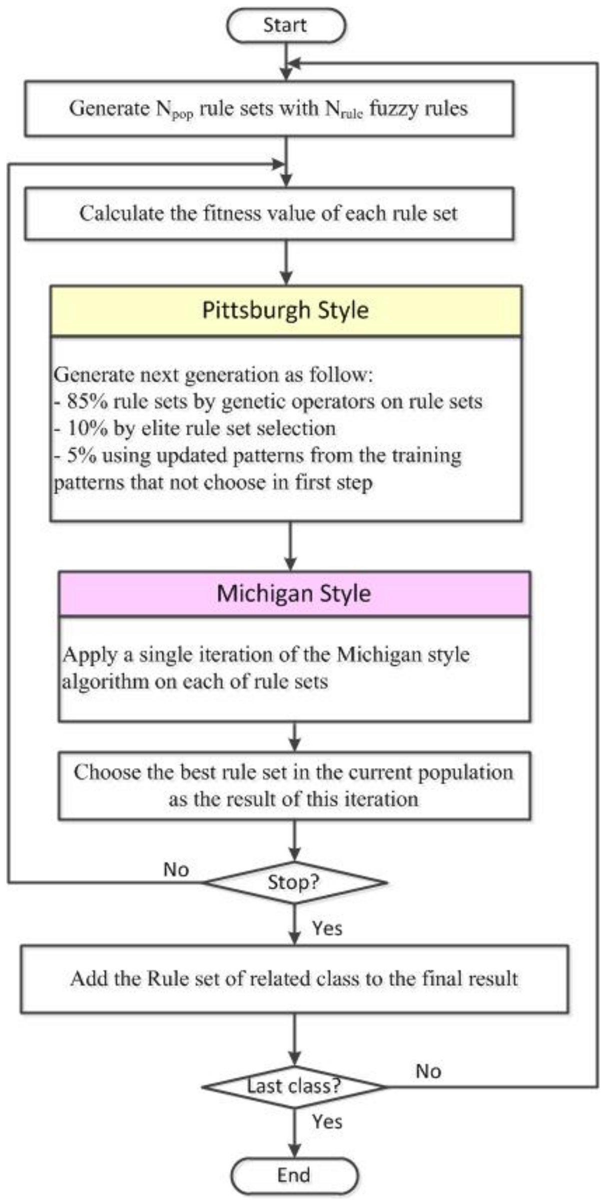

Start
Generate N_{pop} rule sets with N_{rule} fuzzy rules
Calculate the fitness value of each rule set
Pittsburgh Style
Generate next generation as follow:
- 85% rule sets by genetic operators on rule sets
- 10% by elite rule set selection
- 5% using updated patterns from the training patterns that not choose in first step
Michigan Style
Apply a single iteration of the Michigan style algorithm on each of rule sets
Choose the best rule set in the current population as the result of this iteration
No
Stop?
Yes
Add the Rule set of related class to the final result
Last class?
No
Yes
End

Figure 12: The basic approach flowchart

3.4.2 Enhanced IduFG

The second approach is an enhanced version of the basic approach. Producing the result for each class in this approach is the same as the previous one, while the formation of the final result has been done in a different way. As Figure 11 indicates, in the basic approach the final result consists of all the rules in rule set of each class, but as Figure 13 shows the final result contains the elite rule of the rule set of each class. Therefore the number of the final rules is equal to the number of a rule set.

Figure 14 illustrates the flowchart of the second approach. The same as the basic approach, the algorithm is running for each class separately and generates a rule set as a result of the algorithm for each class, but against the basic approach, to generate the final result, in this approach, elite rules of these rule sets will be selected to form the final result.

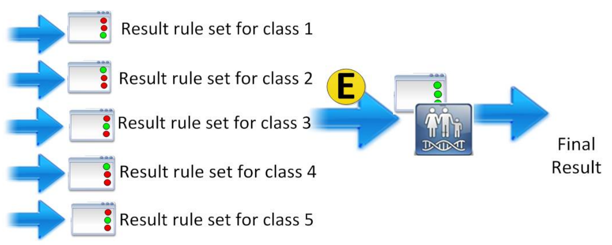

Result rule set for class 1
Result rule set for class 2
Result rule set for class 3
Result rule set for class 4
Result rule set for class 5
E
Final Result

Figure 13: Final result of enhanced approach

4 Simulation Results and Evaluation

In this section, we show the results of the implementation of our proposed algorithm and compare it with one of the best and nearest work proposed in [15]; from now on, we

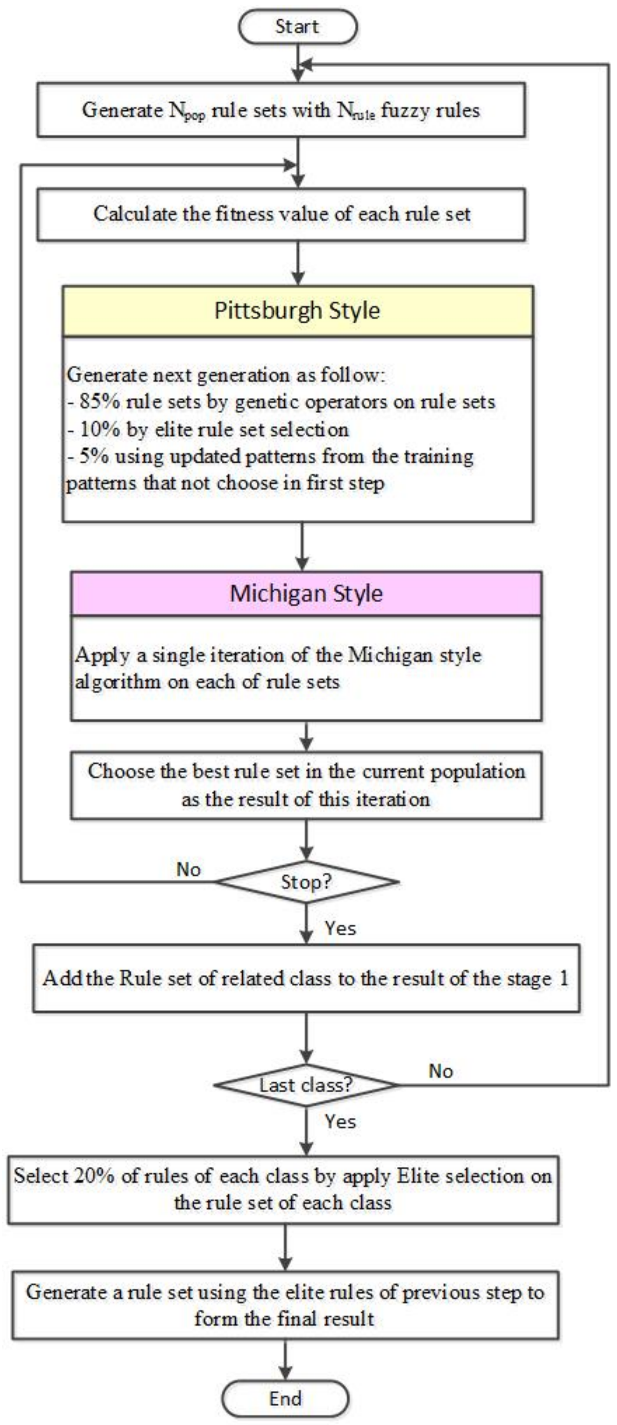

Start
Generate N_{pop} rule sets with N_{rule} fuzzy rules
Calculate the fitness value of each rule set
Pittsburgh Style
Generate next generation as follow:
- 85% rule sets by genetic operators on rule sets
- 10% by elite rule set selection
- 5% using updated patterns from the training patterns that not choose in first step
Michigan Style
Apply a single iteration of the Michigan style algorithm on each of rule sets
Choose the best rule set in the current population as the result of this iteration
No
Stop?
Yes
Add the Rule set of related class to the result of the stage 1
Last class?
No
Yes
Select 20% of rules of each class by apply Elite selection on the rule set of each class
Generate a rule set using the elite rules of previous step to form the final result
End

Figure 14: The enhanced IDuFG flowchart

name this approach as Hybrid approach in this paper. In this section, first, we discuss about the dataset that we use during our tests. After that, the results of the algorithm simulation will be discussed.

4.1 Dataset

As we mention in previous sections, one of the important challenging points in designing IDS , is dataset. It is important to use a good dataset for the test phase. The dataset contains features of normal and abnormal behavior use for designing and test intrusion detection systems. There are two ways to model IDS. The first is to create your own dataset and the second is using the existing datasets such as Knowledge Discovery and Data Mining (KDD). Many researchers used KDD CUP 1999 dataset (KDD), but Mahoney and Chans [20] criticized the dataset validity. As they claim information in the dataset does not look like a real traffic in many aspects. Based on their analysis, the IDS created by using KDD with low false alarm may generate high false alarm in real environment. The best advantage of using the existing datasets is that we can evaluate our work with others; but actually because none of the existing datasets contains all kind of todays attacks it is recommended to create your own dataset with as many as possible kind of attacks. Also for different websites and web applications we may encounter different attacks, so it is better to gather the normal and abnormal behavior of their users for each one separately.

The IDS we are proposed in this paper use the existing dataset, because it became possible for us to compare our algorithm with others. But we intend to create a web attack dataset in future works. We will start our work with capturing data from a web application in a normal and an attack situation. In the next step we will get the results of this step in form of log files as an input. Then we will extract some features from these logs. According to these features we will create our own dataset.

Previous works that use their own dataset has limitations in accessibility of suitable web application. Also they encounter several administrative limitations. Furthermore the ability of web intrusion multiple classifications cannot be seen in most of them. To do a more weighty work, we will gather the traffic of a real web application with high traffic and use unlimited session time that help us to increase data density and more accuracy. We will test and simulate more attacks on the selected web application to achieve a better traffic.

We design an IDS to protect a specific system, with higher search ability rather than previous works. However, we used an improved version of KDD called NSL-KDD that is available on [2]. We have utilized this dataset, to train and test our algorithm; because it became possible to compare our algorithm with others.

NSL-KDD is a data set for the evaluation of anomaly detection methods. The data set was generated by gathering the network logs. It consists of two parts, train and test. This dataset contains 41 attributes, a field of label that expresses the class of a pattern, and a difficulty level column. Some important attributes are as follow: duration, protocol type, service, source and destination host, error rate, etc. The content of Label column contains twenty one attack types and normal state. The complete list of attributes is available in [6].

We use NSL-KDD data set to be able to compare our approaches with previous works. It is an improved version of KDD cup 99 data set [21], although it has some problems mentioned in [26], because there is no other public data set in this domain, researchers use it as a benchmark dataset to evaluate their intrusion detection method.

NSL-KDD is a data set containing numeric and non-numeric values; each row of the data set relating to the network traffic, is a pattern. Each pattern consists of 41 attributes [6], that represent like $(x_{p1}, \ldots, x_{p41})$, $p = 1, 2, \ldots,$ number of patterns. Before we start using it, we normalized the numerical values of the extracted patterns into numbers in interval [0,1]. In [10] four different schemes of attributes normalization are introduced to preprocess the data for anomaly intrusion detection. As systematical evaluation results show, normalization improves the detection performance and among the introduced schemes, the statistical normalization scheme is the tiptop one for a large dataset. In [7], authors discussed about how to extract fuzzy rules directly from numerical data for the pattern classification.

According to our studies and the subjects mentioned in [11] explicitly, we encounter five classes including four main attack classes and a normal class instead of twenty two classes in NSL-KDD data set. Table 3 shows the detailed attacks in four main classes. Below you can find a brief description of these classes:

1) Normal patterns produced by normal and usual behavior of a user such as visiting a web page, streaming a video and so on.

2) Denial of Service (DoS) class contains attacks that by using normal connections and behavior of a user in a massive amount try to down the system. This kind of attacks makes the system unable to service the authorized users. The patterns with back, land, neptune, pod, smurf, teardrop labels classify as DoS attack. Figure 15 illustrates a sample of DoS attack to a target server.

3) Remote to User (R2L) is a class of attacks that exploit by a bug. These attacks make the remote user to access the account of a local user. This class consists of ftp_write, guess_passwd, imap, multihop, phf, spy, warezclient, warezmaster labels.

4) User to Root (U2R) class uses some vulnerability that makes the user to be able to gain the root access. Buffer_overflow, perl, loadmodule, rootkit labels are in U2R class.

5) Probing attack is any kind of effort for information gathering by scanning the target system or network to find its vulnerabilities. These vulnerabilities help the attacker to intrude the system. Ipsweep, nmap, portsweep, satan are the labels of this class.

In most IDS, all of 41 existing feature in KDD data set is used to detect the intrusion, while some of them are repetitive and impertinent. These useless features lead to a time consuming detection process means the causes lower performance. In [18] S. Lakhina et al introduced a new hybrid approach called PCANNA (Principal Component Analysis Neural Network Algorithm) for feature reduction. These approaches reduce the time and memory usage for intrusion detection. It should be mentioned that, in our proposed approaches we use the same feature selection method.

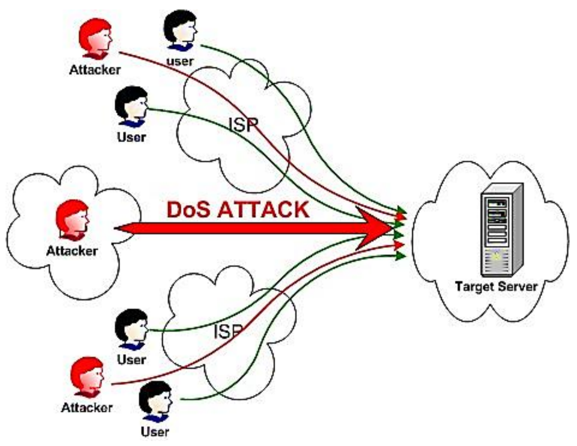


Figure 15: Denial of service attack [3]

Table 3: Different type of attacks in KDD dataset [11]

Four Main Attack Classes	Twenty two Attack Types
Denial of Service (DoS)	Back, land, neptune, pod, smurt, teardrop
Remote to User (R2L)	ftp_write, guess_passwd, imap, multihop, phf, spy, warezclient, warezmaster
User to Root (U2R)	Buffer_overflow, perl, loadmodule, rootkit
Probing	Ipsweep, nmap, portsweep, satan

Before using KDD dataset a set of preprocessing actions should be taken on a dataset to prepare it for use. One of the most important actions is normalization. Although there are some anomaly intrusions detection method that does not normalize the data set before test and train. There are several normalization methods but still discussions on their efficiency exist. In [24] four different schemes of attributes normalization are introduced to preprocess the data for anomaly intrusion detection. As systematical evaluation results show, normalization improves the detection performance and among the introduced schemes, the statistical normalization scheme is the tiptop one for a large dataset. We use this method for normalization in our implementations.

4.2 Parameter Setting

There are several parameters in genetic algorithm such as the population size, the mutation rate, the crossover rate, and so on; that influence on the performance and the result of the algorithm. In this subsection, we are going to introduce them briefly and mention the value that we assign to them in our implementation.

- **Population** Size is one of the important parameters in the genetic algorithm. It says how many individuals are in one generation. If the number of individuals is too few, only a small part of search space is explored and the genetic algorithm has a few possibilities to perform crossover. On the other part, too many individuals, leads the genetic algorithm slow down. We specify 100 rule set that each one contains 20 fuzzy rules as our population.

- **Crossover** Rate indicates the probability of crossover. Simply it states the number of individuals that participate in the crossover. Generally, it should be high, about 0.8-0.95.

- **Mutation** Rate is the probability of alteration of attributes of an individual in the population. It is usually a small value between 0.01-0.1. Of course it could be in 0-1 interval but the mentioned rate is the best average in general.

- W_t is the weight of the correct classification. This value is used in fitness calculation. Actually the number of patterns that classify accurately is multiple by this weight.

- W_f is the weight of the incorrect classification. This value is also used in fitness calculation. Actually the number of patterns that classify erroneously is multiple by this weight and consider as a penalty for the related fuzzy rule.

According to the similar related work, we set the mentioned parameters in our implementation. Table 4 summarizes these parameter values. These parameters are set according to the previous works and experimental results.

Table 4: Parameters value in computer simulation

Parameter	Value
Number of rule sets	100
Number of rules in each rule set	20
Number of Generations	50
Crossover probability	0.9
Mutation probability	0.1
w_t	0.1
w_f	0.9

4.3 Simulation Results

In this section, we test our proposed approaches on NSL KDD; in the following subsections, first we introduce the assessment measures, then we explain the results of the test for each approach.

4.3.1 Assessment Measures

To assess the proposed approaches we use different measures that have been used in most literatures for intrusion detection evaluation [17]; these measures are Recall, Precision, F-measure, and accuracy. The important factors in IDS are maximizing the accuracy while minimizing the False alarms.

Recall is the fraction of Correctly Classified Instances or p(p), to the total number of input patterns that must have been classified correctly. In other words, recall is a fraction of True Positive Rates to the number of all cases that should have been classified as positive. Recall can be defined according to the following equation (TP and FN indicate True Positive and False Negative):

$$Recall = \frac{TP}{TP+FN}.100. \tag{8}$$

Precision is the fraction of true positive instances to all positive instances including those positive instances as classified by the algorithm. In other words, precision is the number of correct results divided by all the results that have been specified by algorithm. Precision then can be seen as a measure of exactness while recall can be seen as a measure of completeness (FP indicates False Positive).

$$Precision = \frac{TP}{TP+FP}.100. \tag{9}$$

F-Measure or F1 is another measure which uses both precision and recall as shown in Figure 16. The value of F measure can show how accurate the algorithm has been, means the higher F shows that an algorithm has been more accurate. The following is the formula for obtaining F score:

$$F_Measure = \frac{2.Recall.Precision}{Precision+Recall}. \tag{10}$$

Accuracy indicates the percentage of closeness of the obtained value to the actual value of the algorithm. It is calculated by the following equation. Figure 17 illustrates the difference between accuracy and precision.

$$Accuracy = \frac{TP+TN}{p(p)+p(n).} \tag{11}$$

4.4 Basic Approach

We compare our basic approach with the Hybrid approach, since it is the nearest work to ours. We find the measures introduced above for each of the approaches and briefly describe them in follow. These measures have been used in literatures to compare and assess many proposed algorithms.

The True Positive Rate of Basic IDuFG is 98.052 and the False Positive Rate is 0.511; while in case of Hybrid approach introduced by Ishibuchi in [14], True Positive Rate is 97.865 and the False Positive Rate is 0.7251. This shows that Basic IDuFG has been better able to report a case as positive where it has actually been positive

Recall
F-Measure
Precision

Figure 16: F-measure definition [17]

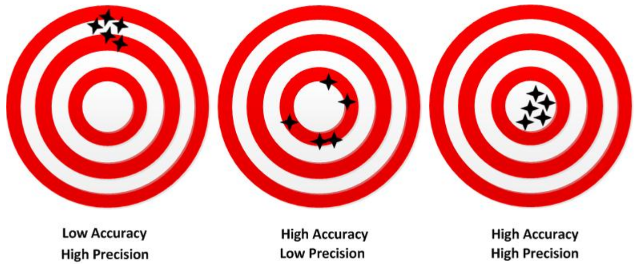


Figure 17: Accuracy and precision

compared to Hybrid approach. It also shows that Basic IDuFG has been less likely to report a case as positive when in fact it has not been positive compared to the Hybrid approach.

The recall rate for Basic IDuFG in the dataset for this experiment is 99.481 and for Hybrid approach is 99.256. According to the definition of recall this means that among all the instances that should have been classified as positive, Basic IDuFG has been able to spot a large number of them as positive while Hybrid approach has not been very successful in differentiating the positive instances.

Since F-measure shows the accuracy of an algorithm, the higher F shows that an algorithm has been more accurate. In this research the obtained F value, for Basic IDuFG and Hybrid approach are 99.089 and 98.420 respectively. This suggests that Basic IDuFG has had a higher accuracy when compared to Hybrid approach. Table 5 shows a comparison between the hybrid GBML that introduce in previous work by Ishibushi and two proposed approaches.

To examine the classification accuracy, first we run the algorithm for each class separately and generate twenty fuzzy rules for each class. Then we check the classification accuracy of this rules for each class separately. The processes of the Classification Accuracy of each class are presented separately in Figures 18 to 20.

The comparison results between hybrid, basic and enhanced approaches are shown in 5. In comparison with hybrid approach that introduced by Ishibuchi, Our basic approach has higher classification accuracy and less false alarm ration while supporting multiple attack classifications, and faster convergence. The enhance approach has a tradeoff between computational time and classification accuracy. In this approach with a little decline in classification accuracy, we reduce the computational time of test phase down to 80%. We can find out that the algorithm behaves differently for the pattern of different classes, but in all of them it converges to the maximum result at most in 50 iterations.

It represent that our proposed algorithm converges to the maximum detection rate on NSL KDD data set faster than the previous hybrid algorithm introduced in [15]. Faster convergence leads to reduction in the run time.

Table 5: Comparison results between hybrid, basic, and enhanced approaches

Measure	Hybrid GBML	Basic IDuFG	Enhanced IDuFG
True Positive	97.865	98.052	97.4
False Negative	0.725	0.511	0.63
Recall	99.256	99.481	99.357
Precision	97.598	98.701	98.54
F-Measure	98.42	99.089	98.701
Classification Accuracy	97.891	98.199	97.437

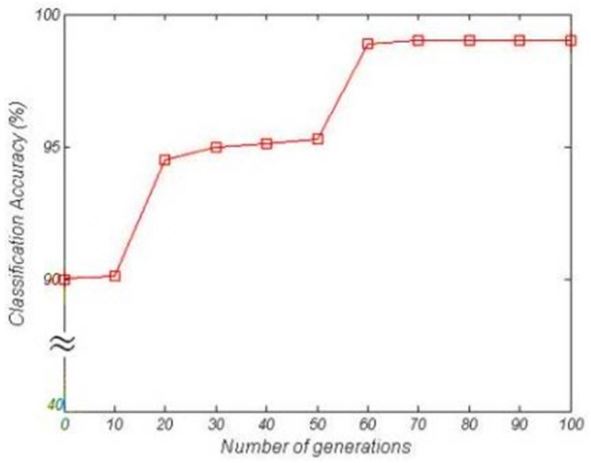


Figure 18: Classification accuracy of basic IDuFG for normal patterns

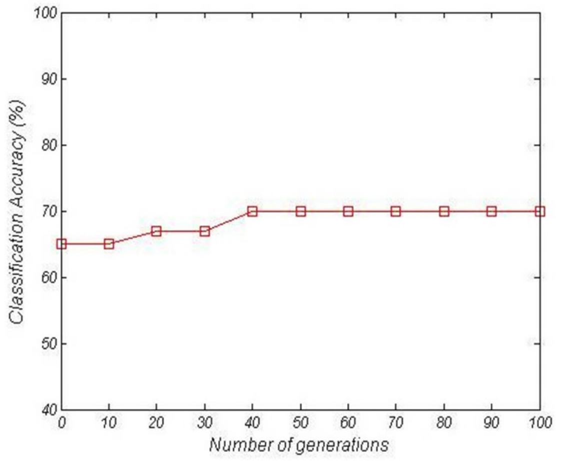


Figure 19: Classification accuracy of basic IDuFG for U2R patterns

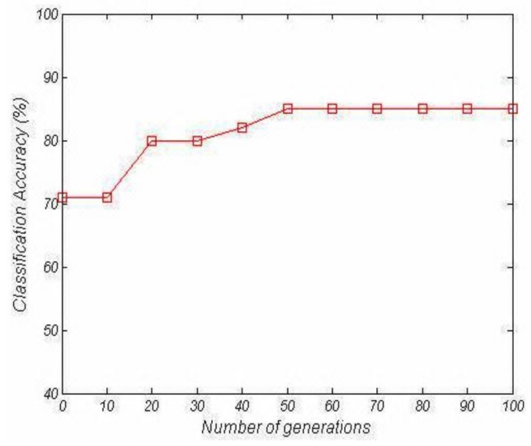


Figure 20: Classification accuracy of basic IDuFG for R2L patterns

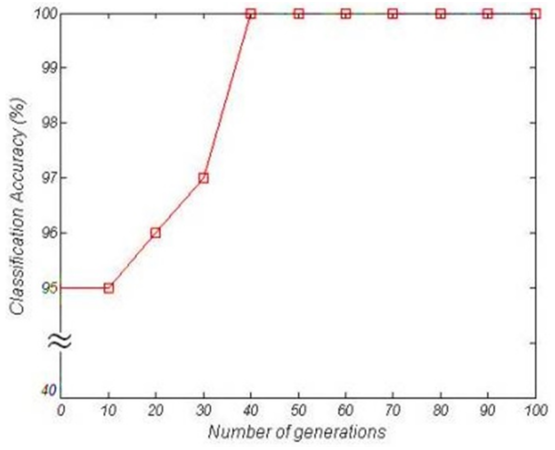


Figure 21: Classification accuracy of basic IDuFG for DOS patterns

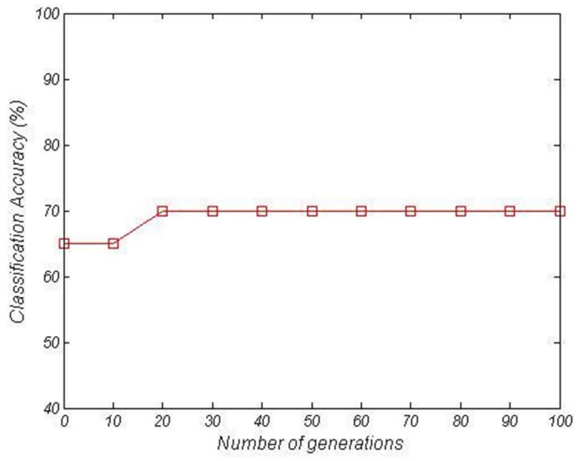


Figure 22: Classification accuracy of basic IDuFG for PRB patterns

We can find out that the algorithm behaves differently for the pattern of different classes, but in all of them it converges to the maximum result at most in 50 iterations.

As it is obvious in these figures, the algorithm behaves differently for the patterns of each class. The reason of this behavior is related to different number of patterns of each class and the relevant feature for each class. For example in DOS class that has the high number of training patterns, we get better classification accuracy.

Table 6: Experimental results of basic approach for each class

Class	Classification Accuracy (%)	False Alarm (%)
Normal	98.5	0.0019
DOS	99.2	0.0047
R2L	89	0.056
U2R	72.3	0.7
PRB	73.8	0.2

Since the number of training patterns of PRB attacks are approximately less than others, we get less classification accuracy as you see in Figure 22. As number of final fuzzy rules is equal to one hundred, in test phase we should compare a pattern with each of them and determine its corresponding class. So the computational time in the test phase is long in comparison with the enhanced approach. The summary of classification accuracy and false alarm rates for each class is shown in Table 6.

First we run the algorithm for each class separately and generate twenty fuzzy rules for each class. Then we check the detection rate of this rules for each class separately. To see the total classification accuracy of these fuzzy rules in basic approach, we check the classification accuracy of these rules for the test patterns of all classes. An average experimental result of over 20 runs is shown in Figure 23 for all classes together. It represents that our proposed algorithm converges to the maximum classification accuracy on NSL KDD data set faster than the hybrid algorithm introduced by [15]. Faster convergence leads to reduction in the run time.

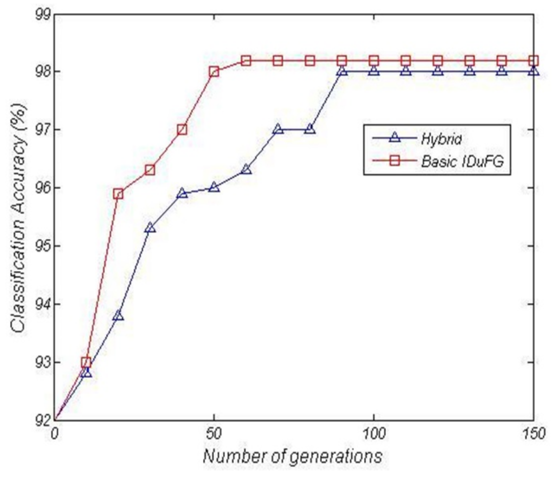


Figure 23: Total classification accuracy of basic IDuFG and hybrid approach

4.5 Enhanced Approach

In order to reduce the computational time in the test phase, we reduce the number of fuzzy rules in the obtained population. As we mentioned above we achieve one hundred fuzzy rules. In the second approach, we decrease the number of fuzzy rules by elite selection of the rule of each class. We select the rules as their certainty grade, the fuzzy rules with the highest certainty is selected in each class. Therefore we have twenty fuzzy rules with the highest certainty grades. The result of testing these rules is shown in Figure 24.

As the experimental results show, although the computational time in the test phase reduces because of the reduction of the fuzzy rules number, the classification accuracy is also reduces a little bit. This is because of the fact that we omit several rules with less certainty grade, but they were able to detect some patterns but not the existing rules. Enhanced IDuFG approach with a little decline of the classification accuracy the computational time in the test phase of the algorithm reduces about 20% of the needed time for the basic approach. Comparison of computational time of these two approaches is shown in Figure 24.

The computational time in the test phase is directly depends on the number of rules. In this approach, it deceases up to 80%, because the number of fuzzy rules is decreased up to 80%. The result is shown in Figure 25. As the test phase time is a small part of the total time (including test and train phase), it influences on the total computational time less than the test phase computational time.

In enhanced IDuFG approach, with a little decline of the classification accuracy, the computational time in test phase of algorithm reduces about 5% of the time needed for basic approach. The comparison of the total computational time of these two approaches is shown in Figure 26.

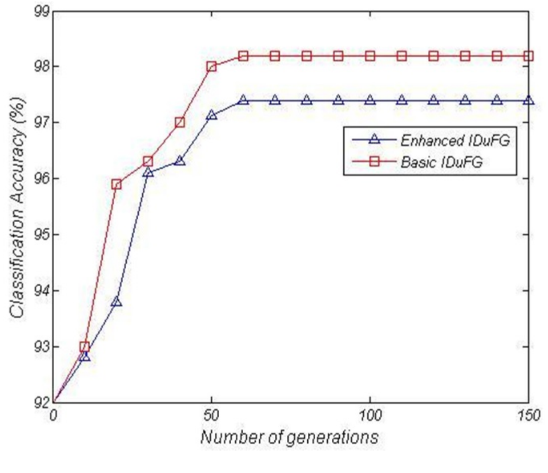


Figure 24: Classification accuracy of basic and enhanced IDuFG

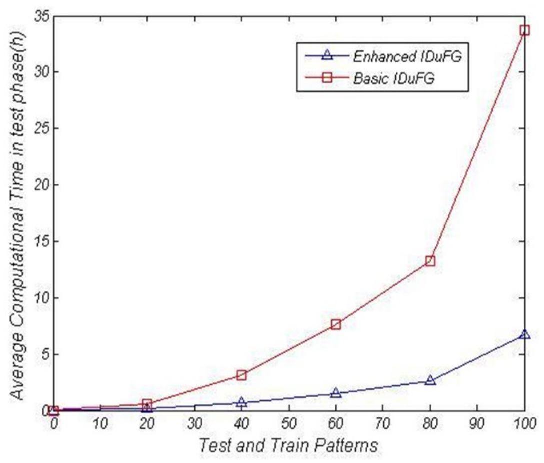


Figure 25: Test phase computational time reduction

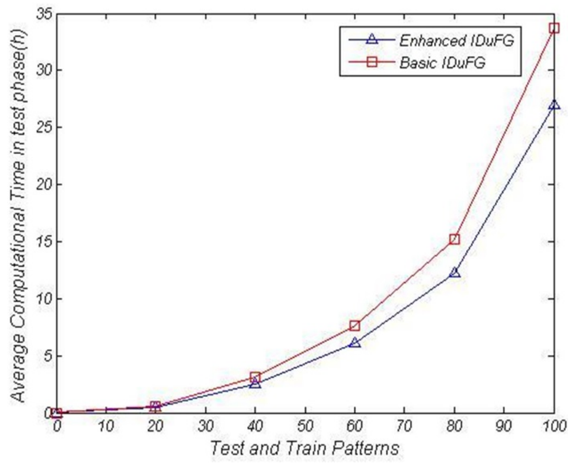


Figure 26: Total computational time reduction

5 Conclusions

Our Proposed system called IDuFG, is an IDS which uses hybrid Fuzzy Genetic algorithm. It supports multiple attack classifications. Two proposed approaches of its algorithm are basic and enhanced. The efficiency of algorithm is kept by using Elitism concept while try to improve it by using more random patterns. The number of generated rules in the enhanced approach is 20% of the basic approach rules. With a little decline of classification accuracy the computational time in the test phase is shorter than the basic approach. So the number of generated rules in the enhanced approach is 80% less than the basic approach, and as a result the computational time of the test phase reduces 80% while the total computational time will be reduced about 5%. With a little decline of the classification accuracy the computational time in the test phase is shorter than the basic approaches.

Finally, as future work we plan to create a dataset consisting of web normal and attack traffic and testing the proposed fuzzy GBML algorithm on the web traffic dataset. Also Utilize Immune systems approach to improve the performance of web anomaly intrusion detection systems.

Acknowledgments

Hereby we express our special thanks to Mr. Iman Khalkhali as Web-based Anomaly Detection Lab (WADL) supervisor for his supports during the tests.

References

[1] http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.

[2] http://iscx.ca/NSL-KDD.

[3] http://www.crazyengineers.com/threads/dos-attack-possibility-on-iran-by-anonymous.63622/.

[4] M. S. Abadeh, J. Habibi, Z. Barzegar, and M. Sergi, "A parallel genetic local search algorithm for intrusion detection in computer networks," *International Journal of Engineering Applications of Artificial Intelligence*, vol. 20, no. 8, pp. 1058–1069, 2007.

[5] M. S. Abadeh, J. Habibi, and E. Soroush, "Induction of fuzzy classification systems using evolutionary aco-based algorithms," in *Proceedings of First Asia International Conference on Modeling and Simulation*, pp. 346–351, Phuket, 2007.

[6] M. S. Abadeh, H. Mohamadi, and J. Habibi, "Design and analysis of genetic fuzzy systems for intrusion detection in computer networks," *International Journal of Expert Systems with Application*, vol. 38, pp. 7067–7075, 2011.

[7] S. Abe and M.-S. Lan, "A method for fuzzy rules extraction directly from numerical data and its application to pattern classification," in *IEEE Transactions on Fuzzy Systems*, vol. 3, pp. 18–28, Ibaraki, 1995.

[8] M.E. Cintra and H. de Arruda Camargo, "Fuzzy rules generation using genetic algorithms with self-adaptive selection," in *IEEE International Conference on Information Reuse and Integration*, pp. 261–266, Las Vegas, 2007.

[9] O. Cordon, F. Herrera, F. Hoffmann, and L. Magdalena, *Genetic Fuzzy Systems*, vol. 19. World Scientific Publishing Publication, 2001.

[10] R. Ensafi, S. Dehghanzadeh, and M. R. Akbarzadeh, "Optimizing fuzzy k-means for network anomaly detection using pso," in *Proceeding of IEEE International Conference on Computer Systems and Applications*, pp. 686–693, Doha, 2008.

[11] D. M. Farid, N. Harbi, E. Bahri, M. Z. Rahman, and C. M. Rahman, "Attacks classification in adaptive intrusion detection using decision tree," in *World Academy of Science, Engineering and Technology*, pp. 86–90, 2010.

[12] A. Fernandez, S. Garcia, J. Luengo, E. Bernado Mansilla, and F. Herrera, "Genetics-based machine learning for rule induction: State of the art, taxonomy, and comparative study," in *IEEE Transaction on Evolutionary Computation*, pp. 913–941, Spain, 2010.

[13] L. Hui and C. Yonghui, "Research intrusion detection techniques from the perspective of machine learning," in *Proceedings of the Second International Conference on Multimedia and Information Technology*, pp. 166–168, Kaifeng, 2010.

[14] H. Ishibuchi and T. Nakashima, "Improving the performance of fuzzy classifier systems for pattern classification problems with continuous attributes," in *IEEE Transactions on Industrial Electronics*, vol. 46, pp. 1057–1068, 1999.

[15] H. Ishibuchi, T. Yamamoto, and T. Nakashima, "Hybridization of fuzzy gbml approaches for pattern classification problems," in *IEEE Transaction on Systems, Man, and Cybernetics*, vol. 35, pp. 359–365, Japan, 2005.

[16] V. Jazayeri, J. Habibi, M. S. Abadeh, and P. Pirzadeh, "Network attacks detection rules extraction using combination of pittsburgh and michigan," in *11th International CSI Computer Conference*, pp. 24–26, Iran, 2006.

[17] I. Khalkhali, R. Azmi, M. Azimpour, , and M. Khansari, "Host-based web anomaly intrusion detection system, an artificial immune system approach," *International Journal of Computer Science Issues*, vol. 8, no. 5, pp. 14–24, 2011.

[18] S. Lakhina, S. Joseph, and B. Verma, "Feature reduction using principal component analysis for effective anomaly-based intrusion detection on nsl-kdd," *International Journal of Engineering Science and Technology*, vol. 2, no. 6, pp. 1790–1799, 2010.

[19] Y. Li, L. Guo, Z. H. Tian, and T.B. Lu, "A lightweight web server anomaly detection method based on transductive scheme and genetic algorithms," *International Journal of Computer Communication*, vol. 31, pp. 4018–4025, 2008.

[20] M.V. Mahoney and P. K. Chani, "An analysis of the 1999 darpa/lincoln laboratory evaluation data for network anomaly detection," in *Proceedings of Recent Advances in Intrusion Detection Conference*, pp. 220–237, Ottawa, 2003.

[21] J. McHughi, "Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory," in *ACM Transactions on Information and System Security*, vol. 3, p. 262–94, 2000.

[22] Y. Nojima, H. Ishibuchi, and I. Kuwajimai, "Comparison of search ability between genetic fuzzy rule selection and fuzzy genetics-based machine learning," in *Proceedings of International Symposium on Evolving Fuzzy Systems*, pp. 125–130, Ambleside, 2006.

[23] C.H. Tsang, S. Kwong, and H. Wang, "Genetic-fuzzy rule mining approach and evaluation of feature selection techniques for anomaly intrusion detection," *International Journal of Pattern Recognition Society*, pp. 2373–2391, 2007.

[24] W. Wang, X. Zhang, S. Gombault, and S. J. Knapskog, "Attribute normalization in network intrusion

detection," in *10th International Symposium on Pervasive Systems, Algorithms, and Networks*, pp. 448–453, Kaohsiung, 2009.

[25] S.X. Wu and W. BanZhaf, "The use of computational intelligence in intrusion detection systems," *International Journal of Applied Soft Computing*, vol. 10, pp. 1–35, 2009.

[26] H. Zhang, B. Zhang, and F. Wang, "Automatic fuzzy rules generation using fuzzy genetic algorithm," in *Sixth International Conference on Fuzzy Systems and Knowledge Discovery*, pp. 107–112, Tianjin, 2009.

Ghazaleh Javadzadeh received her BS degree in Computer Software engendering from Islamic Azad University- South Tehran Branch, Iran in 2009 and her MS degree in Information Technology from Sharif University of Technology-International Campus, Kish, Iran in 2012. She was also a member of Society Engineering of Computer in Islamic Azad University from 2007 to 2009, and Web-based Anomaly Detection Lab (WADL) in 2012. She was a researcher in Security Research Faculty of ICT Research Institute (ITRC), Tehran, Iran in 2012. She is an instructor at University of Applied Science and Technology, Tehran, Iran. Her main research interests are network and web security, and software security.

Reza Azmi received his BS degree in Electrical Engineering from Amirkabir university of technology, Tehran, Iran in 1990 and his MS and PhD degrees in Electrical Engineering from Tarbiat Modares university, Tehran, Iran in 1993 and 1999 respectively. Since 2001, he has joined Alzahra university, Tehran, Iran. He was an expert member of Image Processing and Multi-Media working groups in ITRC (From 2003 to 2004), Optical Character Recognition working group in supreme council of information and communication technology (From 2006 to 2007) and Security Information Technology and Systems working groups in ITRC (From 2006 to 2008). He was Project Manager and technical member of many industrial projects. Dr Azmi is founder of Operating System Security Lab (OSSL), Medical Image Processing Lab (MIPL), Face and Facial Expression Recognition Lab (FFERL), Web-based Anomaly Detection Lab (WADL) and Optical Character Recognition Lab (OCRL) in Alzahra University. He is currently an Assistant Professor of Computer Engineering at Alzahra University.