An Evolutionary Approach to Generate Fuzzy Anomaly Signatures

Fabio González Jonatan Gómez Madhavi Kaniganti Dipankar Dasgupta

***Abstract*— This paper describes the generation of fuzzy signatures to detect some cyber attacks. This approach is an enhancement to our previous work, which was based on the principle of negative selection for generating anomaly detectors using genetic algorithms. The present work includes a different genetic representation scheme for evolving efficient fuzzy detectors. To determine the performance of the proposed approach, which is named Evolving Fuzzy Rules Detectors (EFR), experiments were conducted with three different data sets. One data set contains wireless data, generated using network simulator (NS2) while the other two data sets are publicly available. Results exhibited that our approach outperformed the previous techniques.**

I. Introduction

There are two main approaches to build intrusion detection systems (**IDS**): misuse detection and anomaly detection. The misuse detection approach uses patterns (called signatures) to detect the presence of a known attack. A signature can be a portion of code, a pattern of behaviour, a sequence of system calls, etc. The anomaly detection approach builds a model of normal behaviour of the system. Any system behaviour that does not match this model is reported as an anomaly.

Each approach has advantages and disadvantages.The main advantage of misuse detection is the effective and efficient detection of known attacks; however, new attacks will be unlikely detected. The strongest point of the anomaly detection approach is its capability in detecting unknown attacks; however, it may report unseen normal behaviour of the system as an anomaly generating high number of false alarms. An ideal intrusion detection system will combine the advantageous characteristics of each approach to generate a high detection rate while keeping a low number of false alarms.

Approaches based on artificial immune systems (**AIS**) have been applied successfully to perform anomaly detection [1], [2], [3], [4]. One major difference with other anomaly detection techniques is that AIS builds a model of the abnormal instead of the normal. This abnormal model is described in terms of a set of anomaly detectors which are generated by an algorithm called negative selection [5]. Depending on the representation used, it is possible to see these detectors as signatures of unknown attacks that can be combined with signatures generated by a misuse detection approach to produce an integrated IDS.

Fabio Gonzalez is with Division of Computer Science, The University of Memphis and National University of Colombia

Jonatan Gomez is with Division of Computer Science, The University of Memphis and National University of Colombia

Madhavi Kaniganti is with Division of Computer Science, The University of Memphis

Dipankar Dasgupta is with Division of Computer Science, The University of Memphis

In [6], an anomaly detection technique, which uses the NS algorithm, was proposed. This approach evolves detectors in the non-self (abnormal) space using a genetic algorithm. The evolved detectors had a hyper-rectangular shape that could be interpreted as rules. The paper demonstrated the usefulness of such a technique to detect a wide variety of intrusive activities on networked computers. This approach was improved in subsequent works by modifying the genetic niching technique [7], and introducing a fuzzy representation of the rule detectors.

The purpose of the present work is to further improve the approach proposed in [6], [7] by introducing a higher level chromosomal representation of the rule detectors based on the structured genetic algorithm [8]. The technique proposed in this paper is used to detect anomalies in network traffic generated by denial of service attacks in a simulated wireless network . Additional experimentation is performed with other intrusion detection data sets publicly available.

II. Previous work

Forrest and her group [5] proposed the *negative selection* (**NS**) algorithm. This algorithm is inspired by the mechanism used by the immune system to train the T-cells to recognize antigens (non-self) and to prevent them from recognizing the body own cells (self). The algorithm can be summarized as follows: first, a set of detectors is generated (e.g. randomly), then, these detectors are compared against the self (normal) set, finally, those detectors that match any self element are discarded, and those that do not are kept.

There are different variations of the algorithm and it was able to solve anomaly detection problems [2], [9], fault detection problems [10], [11], to detect novelties in time series [12], [13], and even applied to function optimization [14].

In [6], a new version of the negative algorithm was proposed. The main differences with respect to the negative selection algorithm of Forrest et al. [5] are:

- The elements of self/non-self space are represented by n-dimensional real vectors.
- The detectors correspond to hyper-rectangles in $\mathbb{R}^n$ and have a high level representation as rules.
- The detectors are evolved using a genetic algorithm that maximizes the covering of the non-self space while minimizing the matching of self points. A niching technique is used in order to evolve multiple detectors that cover cooperatively the non-self space.

Figure 1 shows an example of the type of coverage generated by this algorithm. The basic structure of these detector rules is as follows:

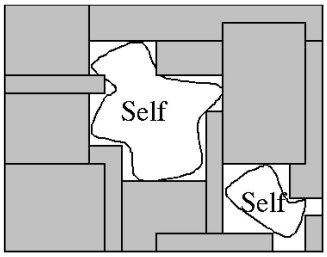

Fig. 1. Approximation of the non-self space by rectangular interval rules.

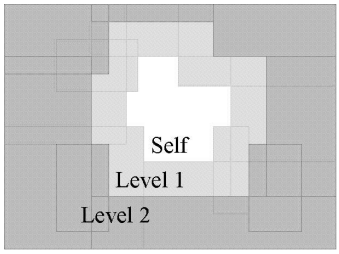

Fig. 2. Two different set of detector rules defi ne two levels of deviation in the non-self space.

$$\begin{array}{lll} R^1\text{: If} & Cond_1 & \text{then } \textbf{non\_self} \\ \vdots & \vdots & \vdots \\ R^k\text{: If} & Cond_k & \text{then } \textbf{non\_self} \end{array}$$

where,

- $Cond_i$ $=x_1 \in [low_1^i, high_1^i]$ and $\ldots$and $x_n \in [low_n^i, high_n^i]$
- $(x_1, ..., x_n)$ is a feature vector
- $[low_i^j, high_i^j]$ specifies the lower and upper values for the feature x_i in the condition part of the rule R^j.

The condition part of each rule defines a hyper-rectangle in the feature space ($[0.0, 1.0]^n$). Then, a set of these rules tries to cover the non-self space with hyper-rectangles. For the case $n = 2$, the condition part of a rule represents a rectangle. Figure 1 illustrates an example of this kind of cover for $n = 2$.

This work also proposed a mechanism that allows to estimate the level of deviation from the normal. The non-self space is further divided in different levels of deviation. In Figure 2, these levels of deviation are shown as concentric regions around the self zones. The genetic algorithm is run as many times as deviation levels are needed. The difference between each run is determined by a variability parameter which specify the degree of variation from the normal set.

In [7] an improvement of this algorithm was proposed. Specifically, it used a different niching technique to generate the rule detectors. The initial algorithm used a sequential niching technique, whereas the new one used deterministic crowding, which proved to be more efficient on generating good anomaly detector rules.

III. Generating anomaly detection signatures

The technique used in this paper is based on the one described in the previous section [6], [7]. Also, it incorporates the improvement proposed in [Fuzzy03 paper] which adds fuzzy logic to the representation of the rule detectors. The use of fuzzy rules improves the accuracy of the method and produces a measure of deviation from the normal that does not need a discrete division of the non-self space. The present work introduces a new chromosomal representation of the rule detectors based on the *structured genetic algorithm* (**StGA**) proposed by Dasgupta in [8]. This new chromosomal representation improves the convergence of the algorithm making it more efficient without sacrificing accuracy.

A. Anomaly detection with fuzzy rules

The self/non-self space corresponds to $[0.0, 1.0]^n$; therefore, an element x in this space is represented by a vector $(x_1, \ldots, x_n)$ where $x_i \in [0.0, 1.0]$. A fuzzy detection rule has the following structure:

$$\textbf{If } x_1 \in T_1 \wedge \ldots x_n \in T_n \textbf{ then } \text{non\_self},$$

where

$(x_1, \ldots x_n)$: element of the self/non-self space being evaluated
T_i: fuzzy set
$\wedge$: fuzzy conjunction operator (in our case, $\min()$)

The fuzzy set T_i is defined by a combination of basic fuzzy sets (linguistic values). Given a set of linguistic values $S = \{S_1, \ldots, S_m\}$ and a subset $\widehat{T}_i \subseteq S$ associated to each fuzzy set T_i,

$$T_i = \bigcup_{S_j \in \widehat{T}_i} S_j,$$

where $\bigcup$ corresponds to a fuzzy disjunction operator. We used the addition operator defined as follows:

$$\mu_{A \cup B}(x) = \min\{\mu_A(x) + \mu_B(x), 1\}.$$

The following is an example of a fuzzy detector rule in a self/non-self space with dimension $n = 3$ and using linguistic values $S = \{S,\ M,\ L\}$:

$$\textbf{If } x_1 \in S \wedge x_2 \in (S \cup M) \wedge x_3 \in (M \cup L) \textbf{ then } \text{non\_self}$$

In our experiments, the basic fuzzy sets correspond to a fuzzy division of the real interval $[0.0, 1.0]$ using triangular and trapezoidal fuzzy membership functions. Figure 3 shows an example of such a division using five basic fuzzy sets representing the linguistic values *Low*, *Medium-Low*, *Medium*, *Medium-High* and *High*.

Given a set of rules:

$$\begin{array}{lll} R^1\text{: } \textbf{If} & Cond_1 & \textbf{then } \text{non\_self} \\ \vdots & \vdots & \vdots \\ R^k\text{: } \textbf{If} & Cond_k & \textbf{then } \text{non\_self} \end{array},$$

where $Cond_i$ corresponds to the condition part of the rule R^i, the abnormality degree of a sample x is defined by

$$\mu_{\text{non\_self}}(x) = \max_{i=1,\ldots m} \{Cond_i(x)\},$$

where $Cond_i(x)$ represents the fuzzy true value produced by the evaluation of $Cond_i$ in x. $\mu_{\text{non\_self}}(x)$ represents the degree of membership of x to the non-self set; thus, a value close to zero means that x is normal and a value close to 1 indicates that it is abnormal.

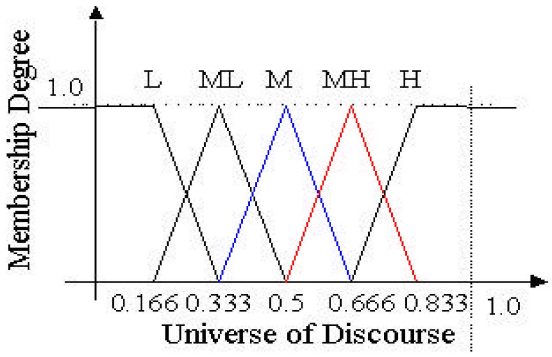

Fig. 3. Partition of the interval [0.0,1.0] in basic fuzzy sets.

```
Initialize population with random individuals;
for j = 1 to numGenerations
  for k = 1 to population_size/2
    Select two individuals with uniform probability
     and without replacement;
    Apply crossover to generate a child;
    Mutate the child;
    if dist(child,parent1) < dist(child,parent2)
      and fitness(child) > fitness(parent1)
       Substitute parent1 with child;
    elseif dist(child,parent1) ≥ dist(child,parent2)
      and fitness(child) > fitness(parent2)
       Substitute parent2 with child;
    endif
  endfor
endfor
Extract the best individuals from the population;
```

Fig. 4. Genetic algorithm to evolve fuzzy rule detectors

B. Evolving fuzzy detector rules

In our previous work [6], we used a genetic algorithm (GA) combined with a niching technique to evolve a set of detector rules that cover cooperatively the non-self space. In the present work, we use the same algorithm, but using deterministic crowding [15] as niching technique since it was shown to perform better than sequential niching [16], as it was demonstrated in [7].

The input to the GA is a set of n-dimensional feature vectors $Self = \{x^1, ..., x^m\}$, which represents samples of normal behavior, the population size and the number of generations. The algorithm is shown in Figure 4.

1) Chromosome representation: For codifying the chromosome we chose a structured representation proposed in [8], the Structured Genetic Algorithm (**StGA**). The StGA utilizes a multi-layered structure that provides a gene activation mechanism and allows the codification of redundant genetic material. The main reason to use this representation is shown in the experimentation section below where the stGA speeds up the GA convergence without sacrificing the quality of the results. Thanks to the fact that a simple change in a high level structure

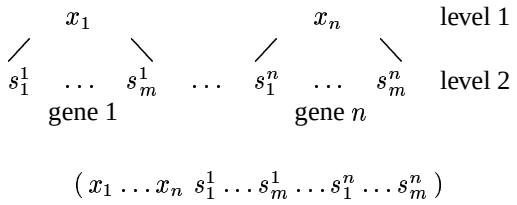

Fig. 5. Structure of the chromosome representing the condition part of a rule. Each gene represents an atomic condition $x_i \in T_i$ and each bit s_j^i is 'on' if and only if the corresponding basic fuzzy set S_j is part of the composite fuzzy set T_j.

can accomplish a phenotypic change that will need a sequence of many random changes in a conventional GA. This characteristic is specially useful for us, given the high dimensionality of the search space and the large size of data sets, which are typical in computer security applications.

Each individual (chromosome) in the genetic algorithm represents the condition part of a rule, since the consequent part is the same for all the rules (the sample belongs to non-self). We used a two-level stGA representation. The high level bits correspond to the features, that is, the length of the high level portion of the chromosome is same as the dimension of the feature space, n. Each bit in the high level portion of the chromosome will tell whether a feature is considered in the condition or not. The low level portion of the chromosome represents the list of atomic conditions that constitute the condition part of the rule.

According to Section III-A, a condition is a conjunction of atomic conditions. Each atomic condition, $x_i \in T_i$, corresponds to a gene in the chromosome that is represented by a sequence $(s_1^i, \ldots, s_m^i)$ of bits, where $m = |S|$ (the size of the set of linguistic values) and $s_j^i = 1$ if and only if $S_j \subseteq T_i$. That is, the bit s_j^i is 'on' if and only if the corresponding basic fuzzy set S_j is part of the composite fuzzy set T_j.

Figure 5 shows the structure of the chromosome which is $n + n \times m$ bits long (n is the dimension of the space and m is the number of basic fuzzy sets).

2) Fitness Evaluation: The fitness of a rule R^i is calculated taking into account the following two factors:

- The fuzzy true value produced when the condition part of the rule, $Cond_i$, is evaluated for each element x from the self set: is a number of elements in the training set S, that belongs to the subspace represented by the rule:

$$selfCovering(R) = \frac{\sum_{x \in Self} Cond_i(x)}{|Self|}$$

- The fuzzy measure of the volume of the subspace represented by the rule:

$$volume(R) = \prod_{i=1}^{n} measure(T_i),$$

where $measure(T_i)$ corresponds to the area under the membership function of the fuzzy set T_i.

The fitness is defined as:

$$fitness(R) = C \cdot (1 - selfCovering(R)) + (1 - C) \cdot volume(R),$$

TABLE I
DATA SETS USED FOR EXPERIMENTATION

Data Set	Training	Testing	
		Normal	Abnormal
Wireless	1160	958	202
Darpa 99	4000	5136	56
KDD-Cup 99	76222	19056	396745

where C, $0 \leq C \leq 1$, is a coefficient that determines the amount of penalization that a rule suffers if it covers normal samples. The closer the coefficient to 1 the higher the penalization. In our experimentation, we used values between 0.8 and 0.9.

3) Individual's Distance Calculation: A good measure of distance between individuals is important for deterministic crowding niching, since it allows the algorithm to replace individuals with closer individuals. This allows the algorithm to preserve and form niches.

In this work we use the Hamming distance, because there is a strong relation between each single bit in the chromosome with a single fuzzy set of some particular attribute of the search space. For example, if the s_i^j bit (see Figure 5), in both parent and child fuzzy rule detectors is set to one, both individuals include the atomic sentence $x_i \in s_j$, i.e., they use the jth fuzzy set to cover some part of the ith attribute. Then, the more number of bits the parent and the child have in common, the more area they will cover in common.

IV. EXPERIMENTATION

In order to determine the performance of the proposed approach (Evolving Fuzzy Rules Detectors - **EFR**), experiments were conducted with three different data sets as shown in table I. For determining the scalability of the proposed approach, each individual performs a random sampling of the training set. The size of the sampling was fixed to 400 data elements. Also, two different algorithms were tested in order to compare the performance of the proposed approach: Evolving Rule Detectors (**ERD**), a non fuzzy method as explained in section II, and Parallel Hill Climbing of Fuzzy Rules Detectors (**PHC**), which is an optimization algorithm based on random mutations of potential solutions population. The algorithms were run 1000 iterations with a population size of 200 individuals. The mutation probability for the ERD algorithm was fixed to 0.1 and the ERD was run four times, each time with a different level of deviation (0.1, 0.2, 0.3, and 0.4). The crisp detectors (hyper rectangles) generated by each run are combined to define the final set of detectors produced by the ERD.

There are two elements that define the cost function of an anomaly detection system: the false alarm rate (**FA**), the system produces an alarm in normal conditions, and the detection rate (**DR**), the system detects an attack. A good intrusion detection system is one that has low FA and high DR. In order to compare the performance of the proposed approach we generated a ROC curve [17] for each of the algorithm tested. Also the reported DR is the detection rate obtained for each algorithm when the FA was fixed to 3%.

TABLE II
TOPOLOGY SETTINGS USED FOR THE INFRASTRUCTURE EXPERIMENTATION

Attribute	Value
X Range	670
Y Range	670
Ad-hoc Routing	Destination-Sequenced Distance-Vector
Number of Mobile Nodes	3
Number of Wired Nodes	2
Number of Base Stations	1
Stop time for the simulation	600 seconds
Mac Protocol	802.11

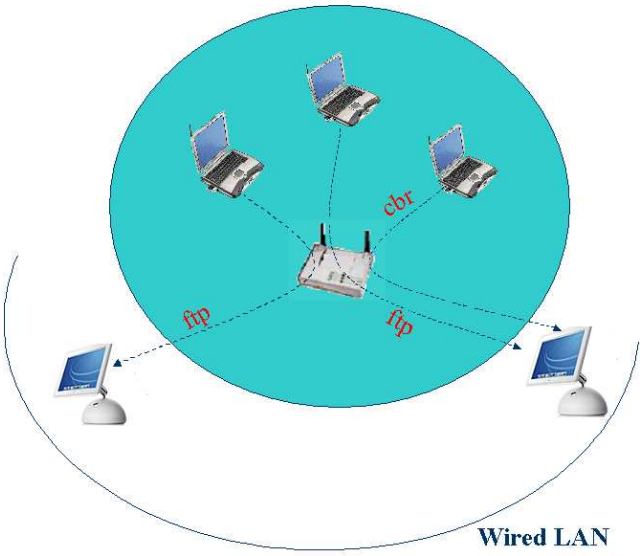

Fig. 6. Connection topology of the infrastructure network.

A. Wireless data set

We used Network Simulator 2(NS2) [NS2 reference] to develop a simulation environment for an infrastructure wireless network[1]. We have developed some intrusions using NS2 (details explained later). Then these anomalies (intrusions) were detected using the proposed approach.

1) Wireless Infrastructure Network:: The environment has a base station (access point) and three wireless nodes communicating with the base station. Also the base station is attached to two other wired nodes. Wireless node 1 sends FTP traffic using TCP to wired node 1, here wireless node acts as the source and the wired node acts as the sink. Similarly the second wireless node sends FTP traffic to second wired node. Lastly the third wired node sends CBR traffic using UDP to the third mobile node. Topology settings are shown in Table II.

Traffic flows among different nodes is shown diagrammatically in Figure 6. Wireless nodes are represented as WN's and

[1]In an infrastructure wireless network the mobile nodes communicate with the wired network with the help of base stations.

the WiredNodes are represented as N's and Traffic among the nodes is represented with dotted lines in Figure 6. FTP traffic flows between WN1 and N1 and also between WN2 and N2 where WN1, WN2 are the TCP agents and N1, N2 are the TCP Sink agent. CBR traffic flows between WN3 and W2 where WN3 is the UDP agent and W2 is the null agent.

a) Attack Method: Access point spoofing: An attacker sets up his or her own access point, known as a rogue, near the target network or in a public place where the victim might believe that wireless Internet access is available. If the rogue access point's signal is stronger than the signal of the real access point, then in many cases, the victim's computer will connect to the rogue access point. Then, the attacker can wait for the victim to type in passwords or inject attack code into the information flow to compromise the victim's computer.
Main goal of generating this attack is to allow Snooping, password acquisition, identity theft, network access on the WLANs.

Here in our experiment we followed a similar approach to generate the attack. We introduced a second malicious base station near the vicinity of the first base station. Also the malicious base station has higher signal strength than the actual one. In such a situation one of our mobile node (node 3) communicates with malicious base station. This resulted in access point spoofing. We have monitored parameter values like the packet size and number of received packets occurring at the legitimate base station both in the normal conditions and during the attack period. Then once the data is collected we were able to detect these anomalies based on the fuzzy rules.

Description of the features collected is shown in Table III below.

The attack is generated between 200 to 250 seconds and again between 400 to 450 seconds. The attack is generated by introducing a malicious base station using higher signal strength than the legal (original) base station thus spoofing the base station.

2) Results and analysis: The results shown here correspond to the experimentation performed using the Infrastructure Data Set. The experiments were conducted using five techniques:

- Evolving Fuzzy Rules (EFR)
- Evolving Rule Detectors (ERD)
- Parallel Hill Climbing (PHC)
- Structure GA using PHC (SGA PHC)
- Structure GA using EFR (SGA EFR)

Experimentation was performed using these techniques varying the number of iterations.

Figure 7(a) shows the ROC Curve generated by EFR when the number of iterations was varied (between 25 and 200). It is clear that the performance of EFR increases when the number of iterations increases too.

Figure 7(b) is similar to the one in Figure 7(a) but using a different technique, Evolving Rule Detectors (ERD).

It can be observed that the ERD is very sensitive to the number of iterations and to the initial population. For example, when the number of iterations is 25 the performance of ERD is worst than with 50 iterations. Also when the number of iterations is high (150) the performance was not good. The results indicate that a good value for the number of iterations is between 50 and 100. It can be due to the fact that ERD generates a huge number of detectors.

Figure 7(c) shows the ROC Curve for stGA using PHC varying the number of iterations. It can be observed that stGA with PHC is sensitive to the number of iterations. The performance is worse when the number of iterations is 25, slightly better when it is 50. The performance is good and almost the same when the number of iterations falls between 75 and 200.

Figure 7(d) shows the ROC Curve for stGA varying the number of iterations. It can be observed that stGA is almost not sensitive to the number of iterations. Only when the number of iterations is big (200) the efficiency of the StGA decreases a little. Clearly, the StGA is speeding up the evolution process.

In Figure 8(a) we can see the performance of all the five techniques when the number of iterations was fixed to 150. It is clear that in almost all the techniques, PHC is worst that ERD in only in a small portion of the ROC curve, based on fuzzy rule detectors outperform the crisp version. It is clear that the StGA approaches perform better than the other techniques. It can due to the fact that the StGA speeds up the convergence of the evolutionary process.

In Figure 8(b) we can see the performance of all the five techniques when the number of iterations was fixed to 75. Again, the StGA techniques outperform the other techniques. On the other hand, the crisp version ERD outperforms the GA and the PHC. A possible explanation for this behavior is that the ERD is generating a large number of small detectors that are covering a large part of the non-self space.

But when the iteration size was changed to 25generations, noteworthy changes were observed as shown in Figure 8(d). StGA using EFR outperformed other techniques.

B. KDD Cup 99

This data set is a version of the 1998 DARPA intrusion detection evaluation data set prepared and managed by MIT Lincoln Labs [18]. Experiments were conducted on the ten percent that is available at the University of Irvine Machine Learning repository [2]. Forty-two attributes, that usually characterize network traffic behavior, compose each record of the 10% data set (twenty-two of them numerical). Also, the number of records in the 10% is huge (492021).

1) Experimental settings: We generated a reduced version of the 10% data set including only the numerical attributes, i.e., the categorical attributes were removed from the data set. Therefore, the reduced 10% data set is composed by thirty-three attributes. The attributes were normalized between 0 and 1 using the maximum and minimum values found. An 80% of the normal samples were picked randomly and used as training data set, while the remaining 20% was used along with the abnormal samples as a testing set. Five fuzzy sets were defined for the 33 attributes. For reducing the time complexity of the ERD algorithm, 1% of the normal data set (randomly generated), was used as a training data set.

2) Results and Analysis: The performance obtained by the S-PHC and S-EFR algorithms are almost the same while are better than the performance obtained by ERD, see Figure 9. Table IV compares the performance of the tested algorithms and

[2] `http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`.

TABLE III

SELECTED FEATURES AND THEIR DESCRIPTION FOR INFRASTRUCTURE EXPERIMENTATION

Feature	**Description**
Received Packets Length	The size of the receive packets by node2 was collected during the specified time intervals. This is used to determine if there is any increase in the size of the packets received.
Number of Received Packets	The number of packets receive by the legal base station is collected to monitor the traffic going through it.
Number of TCP packets received	Monitoring the number of TCP packets received by node2.
Number of TCP ACK packets received	Monitoring the number of TCP ACK packets received by node2.
Number of Constant Bit Rate (CBR) packets received	Monitoring the number of CBR packets received by node2.

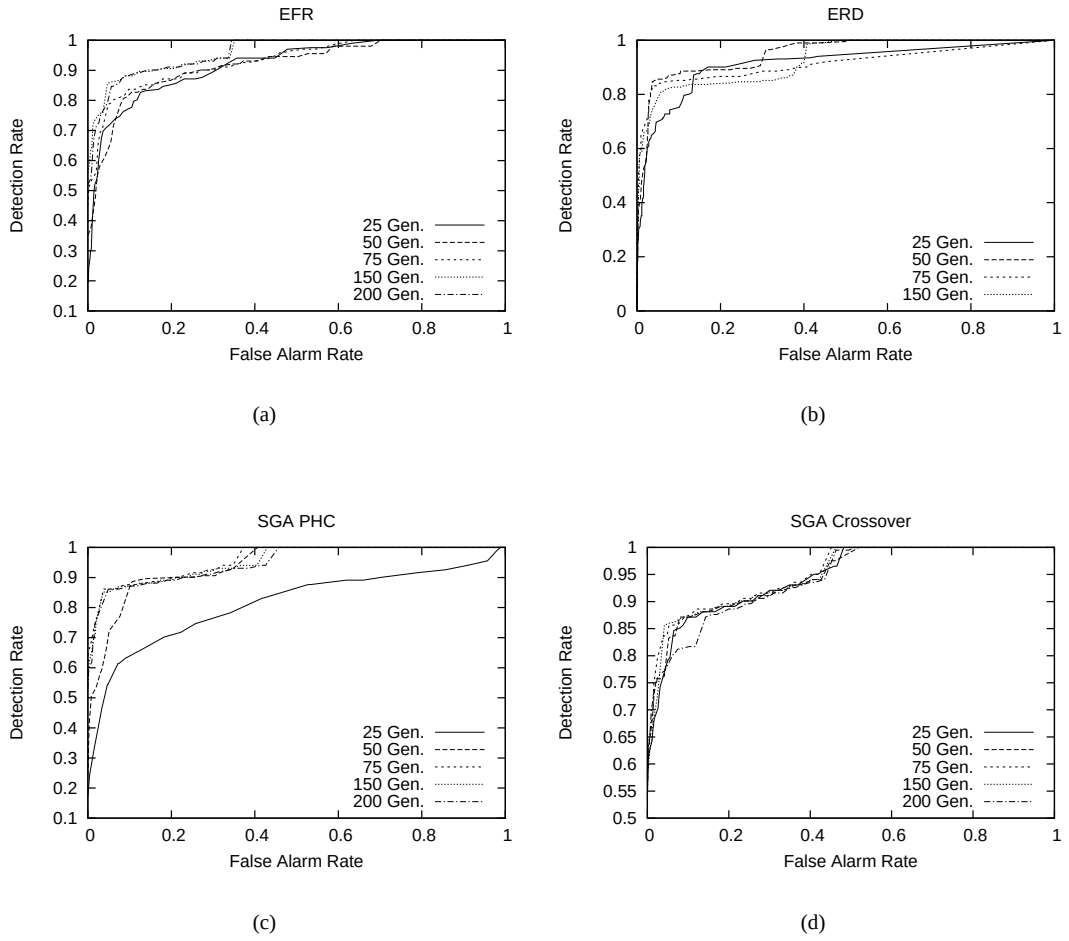

Fig. 7. ROC curves comparing different number of iterations for each of the studied techniques: (a) Evolving fuzzy rule detectors, (b) Evolving rule detectors, (c) StGA with parallel hill climbing, and (d) StGA with fuzzy rules

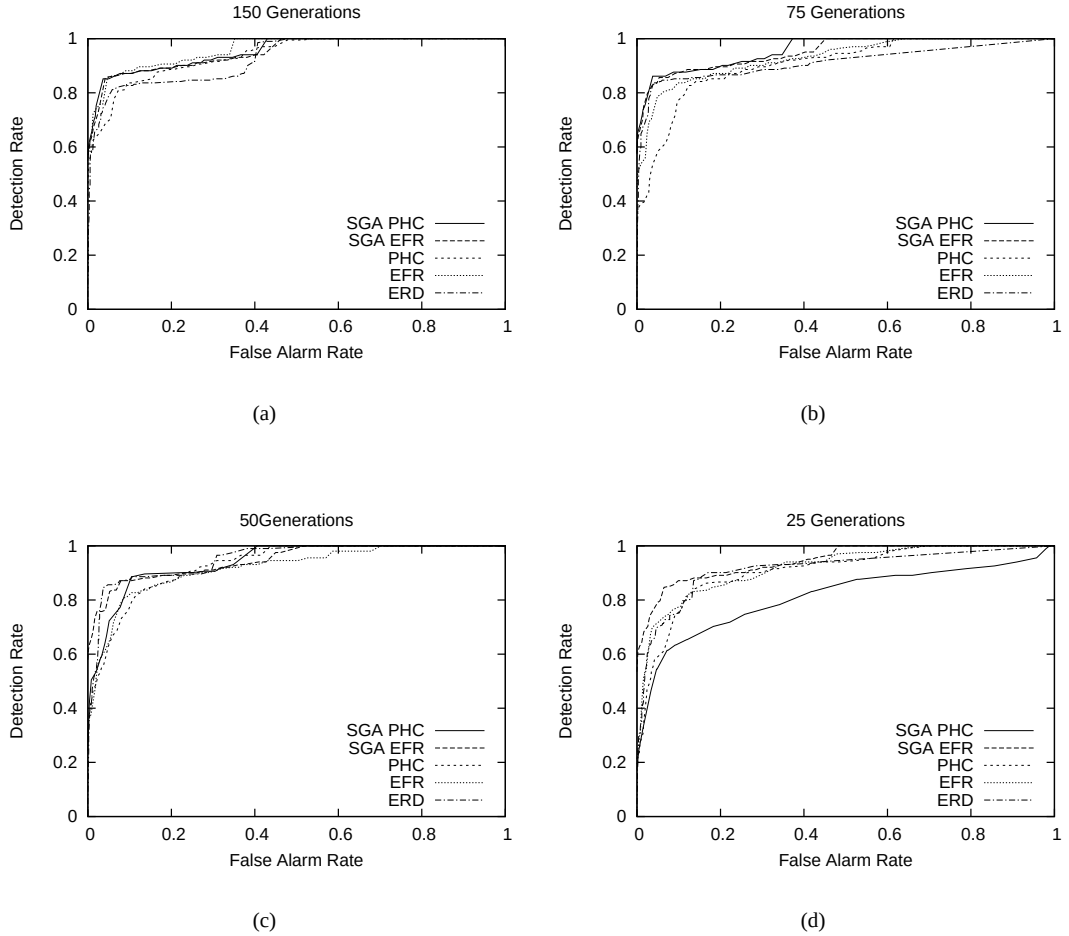

Fig. 8. ROC curves comparing all the five techniques for different number of iterations (generations): (a) 150 iterations, (b) 75 iterations, (c) 50 iterations, and (d) 25 iterations.

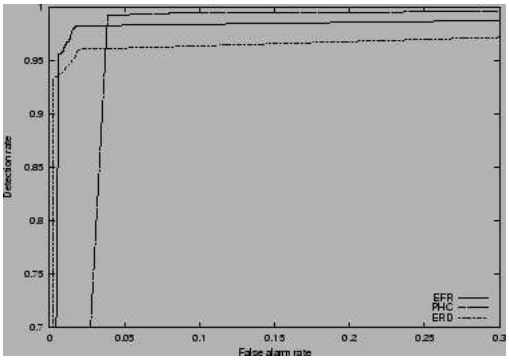

Fig. 9. ROC curves generated by the three algorithms tested with the KDD-Cup 99 data set

TABLE IV
COMPARATIVE PERFORMANCE IN THE KDD CUP 99 PROBLEM

Algorithm	DR%	FA%	# Detectors
S-EFR	**98.22**	**1.9**	14
S-PHC	99.17	3.9	32
ERD	96.02	1.9	699
EFRID[19]	98.95	7.0	-
RIPPER-AA[20]	94.26	2.02	-

some results reported in the literature. The FA-DR reported in table IV is the closest value to the optimal point (0,1). Amazingly, the number of detectors using fuzzyfication is very small compared to the number of detectors using the crisp characterization. It can be due to the high dimensionality of the data set (33 attributes).

According to table IV, the performance of S-EFR is comparable with the performance of approaches reported in the literature and in many cases performs better. For example, when S-EFR is compared with RIPPER-AA the detection rate is almost the same (close to 2%) but S-EFR has a higher DR (4% more abnormal samples detected). Now, compared with the crisp approach (ERD) the performance is also superior (2.2% more abnormal samples detected). Clearly, the fuzzy characterization of the abnormal space reduces the number of false alarm while the detection rate is increased.

Besides the detection rate reached by the S-PHC algorithm is higher than that reached by S-EFR, the false alarm rate is

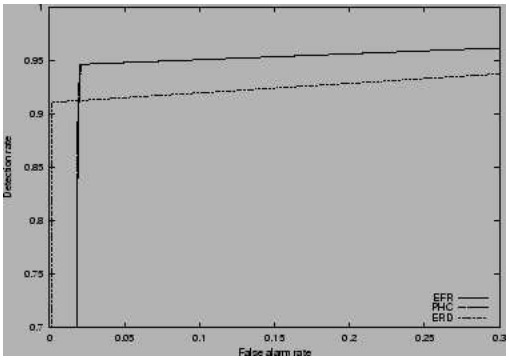

Fig. 10. ROC curves generated by the three algorithms tested with the Darpa 99 data set

also higher (3.9%) than in the S-EFR (1.9%). Also, the number of fuzzy rules detectors generated by S-PHC is large (32) compared with the generated by S-EFR (12).

C. Darpa 99

This data set, was also obtained from the MIT-Lincoln Lab [18]. It represents both normal and abnormal information collected in a test network, where simulated attacks were performed. The data set is composed of network traffic data (tcpdump, inside and outside network traffic), audit data (bsm), and file systems data. We used the outside tcpdump network data for a specific computer (e.g., hostname: marx), and then we applied the tool *tcpstat* to get traffic statistics. The first week's data was used for training (attack free), and the second week's data for testing (this includes some attacks). We only considered the network attacks in our experiments.

1) Experimental Settings: Three parameters were selected (bytes per second, packets per second and ICMP packets per second), to detect some specific type of attacks. These parameters were sampled each minute (using *tcpstat*) and normalized. Because each parameter can be seen as a time series function, the features were extracted using a sliding overlapping window of size $n = 3$. Therefore, two sets of 9-dimensional feature vectors were generated: one as training data set and the other as testing data set. Ten fuzzy sets were defined for each feature extracted.

2) Results and Analysis: The performance reached by the S-PHC and S-EFR algorithms are almost the same while are better than the performance reached by ERD, see Figure 10. These results confirm the hypothesis that a good fuzzyfication of the search space allows fuzzy rule based algorithms to reach a higher performance level than the algorithm based on a crisp characterization of the search space.

Table V compares the performance of the tested algorithms over the Darpa 99 data set. The S-EFR and S-PHC algorithms (both of them based on fuzzy rules detectors), outperformed the ERD algorithm (based on a crisp characterization). Those methods increase the DR in at least 5%. The performance reached by S-EFR and by S-PHC are almost the same, but the number of fuzzy rules detectors generated by S-EFR is lower than the generated by S-PHC. In this way the proposed approach generates a simpler characterization of the abnormal space than S-PHC does.

TABLE V
COMPARATIVE PERFORMANCE IN THE DARPA 99 PROBLEM

Algorithm	DR%	# Detectors
S-EFR	**94.63**	**7**
S-PHC	94.63	9
ERD	89.37	35

V. CONCLUSIONS AND FUTURE WORK

This paper presents a new technique that allows to generate a set of fuzzy rules that characterize the non-self space (abnormal) using as input only self (normal) samples. This work extended a previous work that used crisp rules as detectors. The experiments performed show that the proposed approach performs better than the previous one and is comparable with other results reported in the literature. The following are the main advantages of the new approach:

- It provides a better definition of the boundary between normal and abnormal. The previous approach used a discrete division of the non-self space, whereas the new approach does not need such a division since the fuzzy character of the rules provide a natural estimate of the amount of deviation from normal.
- It shows an improved accuracy on the anomaly detection problem. This can be attributed to the fuzzy representation of the rules which reduce the search space, allowing the evolutionary algorithm to find better solutions.
- It generates a more compact representation of the non-self space by reducing the number of detectors. This is also a consequence of the expressiveness of the fuzzy rules.

Our future work will explore the application of more advanced genetic algorithm representations such as structured GA [**?**] and perform a more extensive testing with other real data sets.

VI. ACKNOWLEDGMENTS

This work was funded by the Defense Advanced Research Projects Agency (no. F30602-00-2-0514) and National Science Foundation (grant no. IIS-0104251).

REFERENCES

[1] D. Dasgupta, *Artifi cial immune systems and their applications*. New York: Springer-Verlag, 1999.
[2] S. Hofmeyr and S. Forrest, "Architecture for an artifi cial immune system," *Evolutionary Computation*, vol. 8, no. 4, pp. 443–473, 2000.
[3] P. Harmer, G. Williams, P.D.and Gnusch, and G. Lamont, "An Artifi cial Immune System Architecture for Computer Security Applications," *IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 252–280, June 2002.
[4] J. Kephart, "A biologically inspired immune system for computers," in *Proceedings of Artifi cial Life*, (Cambridge, MA), pp. 130–139, July 1994.
[5] S. Forrest, A. Perelson, L. Allen, and R. Cherukuri, "Self-nonself discrimination in a computer," in *Proc. IEEE Symp. on Research in Security and Privacy*, 1994.
[6] D. Dagupta and F. González, "An Immunity-Based Technique to Characterize Intrusions in Computer Networks," *IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 281–291, June 2002.
[7] F. González and D. Dasgupta, "An imunogenetic technique to detect anomalies in network traffi c," in *Gecco 2002: proceedings of the genetic and evolutionary computation conference*, (New York), pp. 1081–1088, Morgan Kaufmann Publishers, 9-13 July 2002.

[8] D. Dasgupta and D. McGregor, "A more biologically motivated genetic algorithm: The model and some results," *Cybernetics and Systems: An International Journal*, vol. 25, no. 3, pp. 447–469, 1994.

[9] D. Dasgupta and S. Forrest, "An anomaly detection algorithm inspired by the immune system," in *Artifi cial immune systems and their applications*,, pp. 262–277, Springer-Verlag, Inc., 1999.

[10] D. Dasgupta and S. Forrest, "Tool breakage detection in milling operations using a negative-selection algorithm," Technical Report CS95-5, Department of Computer Science, University of New Mexico, 1995.

[11] A. Tyrrell, "Computer know thy self! : a biological way to look at fault tolerance," in *2nd euromicro/ieee workshop on dependable computing systems*, (Milan), 1999.

[12] D. Dasgupta and S. Forrest, "Novelty detection in time series data using ideas from immunology," in *Proceedings of the International Conference on Intelligent Systems*, pp. 82–87, June 1996.

[13] F. González, D. Dasgupta, and R. Kozma, "Combining Negative Selection and Classifi cation Techniques for Anomaly Detection," in *Proceedings of the Congress on Evolutionary Computation*, (Honolulu, HI), pp. 705–710, IEEE, May 2002.

[14] C. A. C. Coello and N. C. Cortes, "A parallel implementation of the artifi cial immune system to handle constraints in genetic algorithms: preliminary results," in *Proceedings of the 2002 Congress on Evolutionary Computation CEC2002*, (Honolulu, Hawaii), 2002.

[15] S. W. Mahfoud, "Crowding and preselection revisited," in *Parallel problem solving from nature 2* (R. Männer and B. Manderick, eds.), (Amsterdam), pp. 27–36, North-Holland, 1992.

[16] D. Beasley, D. Bull, and R. Martin, "A sequential niche technique for multimodal function optimization," *Evolutionary Computation*, vol. 1, no. 2, pp. 101–125, 1993.

[17] F. Provost, T. Fawcett, and R. Kohavi, "The case against accuracy estimation for comparing induction algorithms," in *Proceedings of 15th international conference on machine learning*, (San Francisco, Ca), pp. 445–453, Morgan Kaufmann, 1998.

[18] "Mit lincoln labs. 1999 darpa intrusion detection evaluation." http://www.ll.mit.edu/IST/ideval/index.html, 1999.

[19] J. Gomez and D. Dasgupta, "Evolving Fuzzy Classifi ers for Intrusion Detection," in *Proceedings of the 2002 IEEE Workshop on Information Assurance*, June 2002.

[20] W. Fan, W. Lee, M. Miller, S. Stolfo, and P. Chan, "Using artifi cial anomalies to detect unknown and known network intrusions," in *Proceedings of the fi rst IEEE International conference on Data Mining*, 2001.