# Detectors Generation using Genetic Algorithm for a Negative Selection Inspired Anomaly Network Intrusion Detection System

Amira Sayed A. Aziz[1,*], Mostafa Salama[2,*], Aboul ella Hassanien[3,*]
Sanaa EL-Ola Hanafi[3]
[1]French University in Egypt, Cairo, Egypt
Email: amira.abdelaziz@scienceegypt.net
[2]British University in Egypt, Cairo, Egypt
Email: mostafa.salama@gmail.com
[3]Cairo University, Faculty of Computers and Information, Cairo, Egypt
[*]Scientific Research Group in Egypt (SRGE)

***Abstract*—This paper presents an approach for detecting network traffic anomalies using detectors generated by a genetic algorithm with deterministic crowding Niching technique. Particularly, the suggested approach is inspired by the negative selection mechanism of the immune system that can detect foreign patterns in the complement (non-self) space. In our paper, we run a number of experiments on the relatively new NSL-KDD data set which was never tested against this algorithm before our work. We run the test using different values for the involved parameters, to find out which values give the best detection rates, so we can give recommendations for future application of the algorithm. Also, Formal Concept Analysis is applied on the generated rules to visualize the relation among attributes. We will show in the results that the algorithm have very good results through the analysis, compared to other machine learning approaches.**

## I. Introduction

ANOMALY detection has been a widely researched problem in several application domains such as system health management, intrusion detection, health-care, bio-informatics, fraud detection, and mechanical fault detection. Traditional anomaly detection techniques analyze each data instance (as a uni-variate or multivariate record) independently. And ignore the sequential aspect of the data. Often, anomalies in sequences can be detected only by analyzing data instances together as a sequence, and hence cannot be detected by traditional anomaly techniques [1]. Gonzalez and Dasgupta in [2] used sequential niching technique with the genetic algorithm to generate the rules. Then, in 2003 [3] they suggested using deterministic-crowding niching technique to limit the crowd by replacing parents with more fitted children. This time, the algorithm gave same results with less number of rules, which is better because the population size won't change. This paper applies an approach for detecting network traffic anomalies using genetic algorithm based intrusion detection system, but without the levels of abnormality. The algorithm is put under investigation to find which values for its parameters can lead to better results, using the relatively new NSL-KDD data set.

The rest of this paper is organized as follows. Section II presents a background of anomaly intrusion detection, artificial immune systems, genetic algorithms and formal concept analysis. Section III gives a description of the applied approach and its phases as well. Section IV shows the experimental results and discusses observations. Finally, section V addresses conclusions and discusses future work.

## II. Background

### A. Anomaly Detection

An Intrusion Detection System (IDS) is a system built to detect outside and inside intruders to an environment by collecting and analyzing its behavior data. Two basic approaches are followed to implement an IDS: Misuse-based and Anomaly-based detection. In a misuse-based IDS, attacks are represented as a pattern or a signature to use for detection. It's very good in detecting known attacks and provide detailed information on the detected ones, but is of little use for unknown attacks. Anomaly-based IDS build a model for a system's normal behavior to use for detection, assuming all deviated activities to be anomalous or intrusions. It is very useful for finding unknown attacks but it has a high false negative or positive rates, beside it needs to be updated with system behavior and can't provide much information on detected attacks. In some IDSs, a hybrid of both techniques is used [4].

Different approaches exist for Anomaly-based Network IDS (A-NIDS), but in general they all consist of the following modules: (1) *Parameterization:* representing the observed instances in some pre-defined form, (2) *Training:* a model is built using the normal or abnormal system behavior. It can be done manually or automatically, and (3) *Detection:* the (parameterized) monitored traffic is searched for anomalous behavior using the system model built through previous stage.

The techniques used to build the system behavioral model can be: statistical, knowledge-based, or machine learning-based. The Genetic Algorithms (GA) is among the machine learning-based techniques. The flexible and robust global search is the main advantage of applying GAs in A-NIDS, where it looks for a solution from multiple directions with no prior knowledge required about the system [5], [6].

## B. Genetic Algorithms

GAs are basically used in IDSs to generate rules used to detect anomalies [6]. They were inspired by the biological evolution (development), natural selection, and genetic recombination. GAs use data as chromosomes that evolve through: selection (usually random selection), cross-over (recombination to produce new chromosomes), and mutation operators. Finally, a fitness function is applied to select the best (highly-fitted) individuals. The process is repeated for a number of generations until reaching the individual (or group of individuals) that closely meet the desired condition [2], [6].

GAs are very promising in the computer security field, especially in IDSs. They have been applied for intrusion detection since the 1990's, and still being used up till the current time. GA is usually used to generate rules for intrusion detection, and they usually take the form *if* {*condition*} *then* {*action*}, where the condition part test the fields of incoming network connections to detect the anomalous ones [6].

## C. Artificial Immune Systems

The Artificial Immune Systems (AIS) were inspired by the Human Immune System which is robust, decentralized, error tolerant, and adaptive. The HIS has different cells with so many different tasks, so the resultant mimic algorithms give differing levels of complexity and can accomplish a range of tasks. There are a number of AIS models used in pattern recognition, fault detection, computer security, and a variety of other applications in the field of science and engineering. Most of these models emphasize on designing and applying computational algorithms and techniques using simplified models of various immunological processes and functionalities [7], [8].

There exists no single algorithm from which all immune algorithms are derived, as AISs are designed using a number of algorithms [9]. The Negative Selection approach (NSA) — which is applied in our paper — explains how T-cells are being selected and their maturation in the system. T-cells are blood cells that belong to a group of white blood cells calles lymphocytes. In the NSA, whenever the T-Cells are produced, they undergo an immaturely period to learn which antigen recognition results in their death. The T-cells need activation to develop the ability to remove pathogens. They are exposed to a comprehensive sample of self antigens, then they are tested against self and non-self antigens to match the non-self ones. If a T-Cell matched a self antigen, it is then removed until they are mature and released to the system [10], [11].

## D. Formal Concept Analysis

Formal Concept Analysis (FCA) is one of the data mining research methods and it has been applied in many fields as medicine. The basic structure of FCA is the formal context which is a binary-relation between a set of objects and a set of attributes. The formal context is based on the ordinary set, whose elements has one of two values, 0 or 1 [12], [13]. A formal concept is defined as a pair(A, B) with $A \subseteq G, B \subseteq M$, intent($A$)=$B$ and extent($B$) = $A$. The set $A$ is called the extent and the set $B$ called the intent of the concept ($A$, $B$). The extent and the intent are derived by two functions, which are defined as:

$$intent(A) = \{m \in M | \forall g \in A : (g, m) \in I\}, A \subseteq G, \quad (1)$$

$$extent(B) = \{g \in G | \forall m \in B : (g, m) \in I\}, B \subseteq M. \quad (2)$$

Usually the attributes of a real life data set are not in a binary form, attributes could be expressed in a many-valued form that are either discrete or continuous values. In that case the many-valued context will take the form $(G, M, V, I)$ which is composed of a set $G$ of objects, a set $M$ of attributes, a set $V$ of attribute values and a ternary-relation $I$ between $G$, $M$ and $V$. Then the many-valued context of each attribute is transformed to a formal concepts, the process of creating single-valued contexts from a many-valued data set is called conceptual scaling. The process of creating a conceptual scale must be performed by using expert knowledge from the domain from which the data is drawn. Often these conceptual scales are created by hand, along with their concept lattice, since they are represented by formal contexts often laid out by hand. Such that we choose a threshold $t$ for each many-valued attribute and replace it by the two one-valued attributes "expression value" [12], [13].

## III. The Proposed Network Anomaly Detection Approach

Genetic Algorithms produce the best individual as a solution, but in an A-NIDS a set of rules is needed - hence, running GA multiple times. The technique used here was originally proposed in [3] to solve this problem using the Deterministic-Crowding Niching technique. Niching algorithms are important research area in evolutionary computation. They basically aim to generate multiple and highly-fit different solutions while slowing down convergence if only one solution is needed. Many Niching techniques exist, but the strengths of the deterministic-crowding are that it requires no additional parameters to those that are already used in a GA, beside that itis fast and simple [14].

The self (normal behavior) individuals are represented in a self space $S$, where each individual is represented as a vector of features of dimension $n$, with the values normalized to the interval [0.0,1.0]. This can be written as $S = x_1, \ldots, x_m$, where $m$ is the number of the self samples. Algorithm (1) shows the main steps of the detectors generation approach.

The final solution is a set of rules, represented as individuals with low and high limits for each dimension, as the conditions used to define the AIS detectors. So, each rule $R$ has a condition part ($x_n \in [low_i, high_i]$), hence a feature vector $x_i$ satisfies a rule $R$ if its hyper-sphere intercepts the

**Algorithm 1** Deetetors generation algorithm

```
1: Initialize population by selecting random individuals from the space $S$.
2: for The specified number of generations do
3:   for The size of the population do
4:     Select two individuals (with uniform probability) as $parent_1$ and $parent_2$.
5:     Apply crossover to produce a new individual ($child$).
6:     Apply mutation to child.
7:     Calculate the distance between $child$ and $parent_1$ as $d_1$, and the distance between $child$ and $parent_2$ as $d_2$.
8:     Calculate the fitness of $child$, $parent_1$, and $parent_2$ as $f$, $f_1$, and $f_2$ respectively.
9:     if ($d_1 < d_2$) and ($f > f_1$) then
10:      replace $parent_1$ with $child$
11:    else
12:      if ($d_2 <= d_1$) and ($f > f_2$) then
13:        Replace $parent_2$ with $child$.
14:      end if
15:    end if
16:  end for
17: end for
18: Extract the best (highly-fitted) individuals as your final solution.
```

hyper-cube represented by the rules defines by its points [3].

To calculate the fitness of an individual (or a rule), two things are to be taken into account: the number of elements in the training sample that can be included in a rule's hyper-cube, calculated as [2], [3]:

$$num\_elements(R) = x^i \in S \text{ and } x^i \in R \tag{3}$$

The volume of the hyper-cube that the rule represents is defined by the following form:

$$volume(R) = \prod_{(i=0)}^{n} (high_i - low_i) \tag{4}$$

Consequently, the fitness is calculated as:

$$fitness(R) = volume(R) - C \times num\_elements(R) \tag{5}$$

where $C$ is a coefficient of sensitivity that represents a penalty if a rule covers anomaly samples. The bigger the $C$ value, the higher the sensitivity - hence the penalty - is. The fitness can take negative values. The same equations are used if you're calculating the fitness of an individual in general. To calculate the distance between two individuals (a child $c$ and a parent $p$), volumes of the hyper-cubes surrounding the individuals (represented by low and high points in each dimension) are used as follows:

$$distance(c,p) = \frac{volume(p) - volume(p \bigcap c)}{volume(p)} \tag{6}$$

The purpose of using the volume is to check how much the child covers of the area of the parent, so this distance measure is not symmetric. To calculate the intersection between two hyper-cubes, compare the intervals of the hyper-cubes in each dimension, which will produce a hyper-rectangle in the end.

## IV. Experimental results and discussion

### A. Data Sets

The experiment was performed on the NSL-KDD data set which was suggested to solve some problems in the KDD Cup'99 data set that is widely used for IDS evaluation. This data set contains less number of records in both the train and the test, which helps researchers to run their experiments on the whole sets instead of only small portions. Hence, the evaluation results will be comparable and consistent [15]. Fifteen parameters (features) were selected to use in our experiment, which have real values that can be used in the approach and can be used to detect basic DoS attacks. These features values are already in the interval [0.0,1.0], and they are listed in Table (I).

TABLE I
Features used in the experiment

| Feature name | Description |
|---|---|
| serror_rate | % Of connections with "SYN" errors (same-host connections) |
| srv_serror_rate | % Of connections with "SYN" errors (same-service connections) |
| rerror_rate | % Of connections with "REJ" errors (same-host connections) |
| srv_rerror_rate | % Of connections with "REJ" errors (same-service connections) |
| same_srv_rate | % Of connections to the same service (same-host connections) |
| diff_srv_rate | % Of connections to different services (same-host connections) |
| srv_diff_host_rate | % Of connections to different hosts (same-service connections) |
| dst_host_same_srv_rate | Same_srv_rate for destination host |
| dst_host_diff_srv_rate | Diff_srv_rate for destination host |
| dst_host_same_src_port_rate | Same_src_port_rate for destination host |
| dst_host_srv_diff_host_rate | Diff_host_rate for destination host |
| dst_host_serror_rate | Serror_rate for destination host |
| dst_host_srv_serror_rate | Srv_error_rate for destination host |
| dst_host_rerror_rate | Rerror_rate for destination host |
| dst_host_srv_rerror_rate | Srv_serror_rate for destination host |

### B. Experiment Settings

We used the self (normal) data only in the training phase to generate best rules that represents the Self profile, as the negative selection approach suggests, then the rules were compared against the test sample. The parameters values used for the genetic algorithm were:

- Population size: 200, 400, 600
- Number of generations: 200, 500, 1000, 2000
- Mutation rate: 0.1

- Sensitivity coefficient: 1.0
- Variability values: 0.05, 0.10, 015, 0.20

Following the NSA, we basically train the algorithm (to generate rules) on self (normal) samples, then use these rules to find non-self (anomalies) which will be the vectors very far from the self rules. To characterize the samples to self or non-self, the characterization function was:

$$\mu_{non\_self(x)} = D(x, Self) = \min\{d(x,s) : s \in Self\} \quad (7)$$

which mean the closer a vector $x$ is to a self point $s$, the less it is a non-self sample.

The distance measure d(x,s), as shown in equation (8), used to characterize the test data was the n-dimensional Euclidean distance, which is:

$$d(x,s) = \sqrt{(x_1 - s_1)^2 + (x_2 - s_2)^2 \ldots (x_n - s_n)^2} \quad (8)$$

## C. Experiment Results

The training phase was held using normal samples extracted from the 20% Train Set to generate intrusion detectors using different population sizes ran for different numbers of generations, one run used for each. Then these rules were used to detect anomalies in both the whole Train Set and the Test Set+. Figures 1 and 2 show the average detection rates and maximum detection rates respectively against different variation levels. The figures show that low variation levels give better results when used with smaller populations, while the bigger the population size, the better the average detection rates of higher variation values used. Following figure 2, we will realize that using higher population size gives better results, especially with less number of generations.

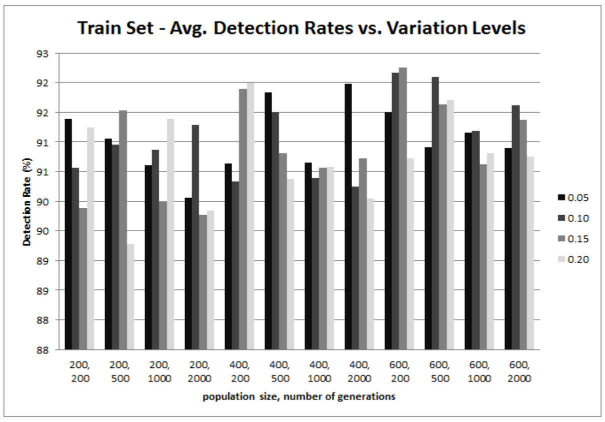

Fig. 1. Train Set Average Detection Rates versus Variation Levels.

As for the Test Set, Figures 3 and 4 show the average and maximum detection rates respectively. we will find that in general detectors generated using lower population sizes give relatively better results with low variation values. But looking at maximum rates in figure 4, we will realize that higher variation values have the best detection rates.

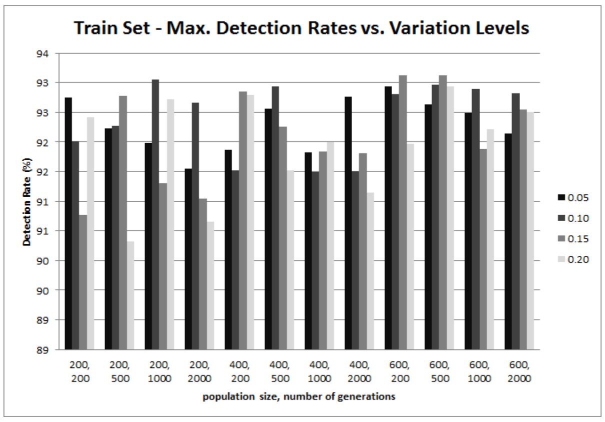

Fig. 2. Train Set Maximum Detection Rates versus Variation Levels.

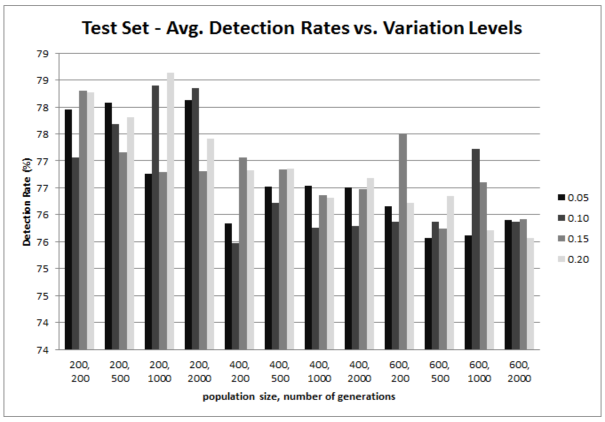

Fig. 3. Test Set Average Detection Rates versus Variation Levels.

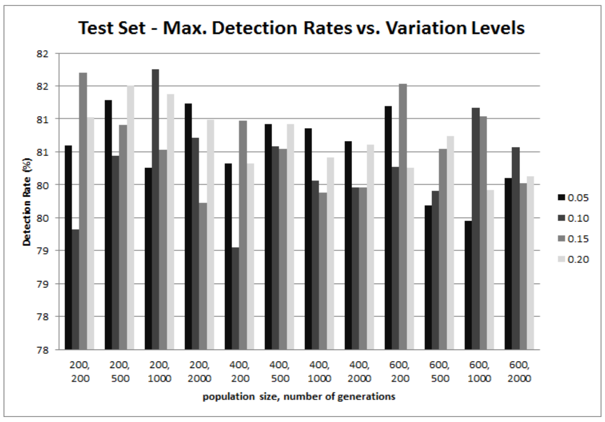

Fig. 4. Test Set Maximum Detection Rates versus Variation Levels.

Coming to different threshold levels used to detect anomalies, figures 5 and 6 show the detection rates versus threshold values (0.2, 0.4, 0.6, 0.8). the detection rates are higher when testing the Train Set with threshold values 0.4, 0.6. Running the intrusion detection process on the Test Set, we will realize that using a threshold of value 0.2 gives the best results.

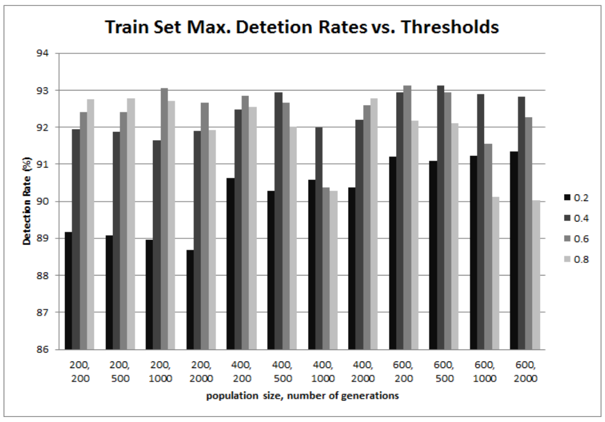

Fig. 5. Train Set Maximum Detection Rates versus Threshold Levels.

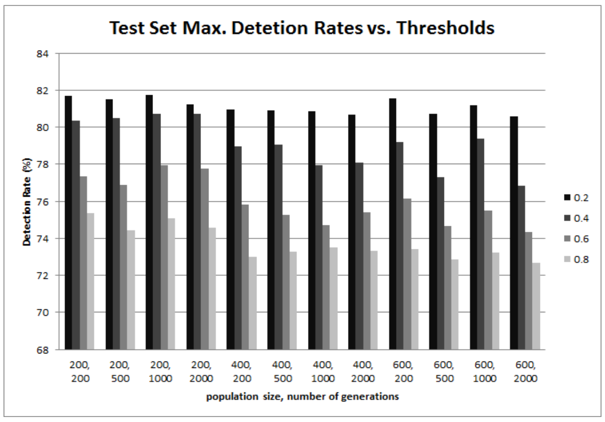

Fig. 6. Test Set Maximum Detection Rates versus Threshold Levels.

## *D. Comparison analysis*

In [16], they ran the machine learning algorithms implemented in the project WEKA [17] against the NSL-KDD Test Set. The detection accuracy is listed in table II along with the suggested algorithm, and it shows that we have very good detection accuracy compared to those approaches used

TABLE II
THE CLASSIFICATION ACCURACY OF KNOWN CLASSIFIERS

| Classifier name | Classification accuracy |
|---|---|
| j47 | 81.05% |
| Naive Bayes | 76.56% |
| NBTree | 82.03% |
| Random Forest | 80.67% |
| Random Tree | 81.59% |
| Multi-layer Perception | 77.41% |
| SVM | 68.52% |
| Suggested Algorithm | 81.76 % |

## *E. Visualization of rules*

Formal Concept Analysis is applied on the rules generated from running the algorithm. Figure 7 shows that attributes A1, A3, and A5 are the most effective in the descrimination between normal and anomalous connections. They correspond to serror_rate, rerror_rate, and same_srv_rate respectively.

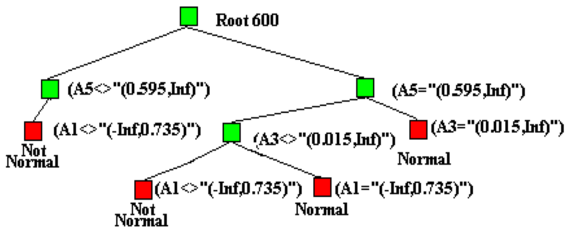

Fig. 7. Visualized generated rules using FCA

## V. CONCLUSIONS AND FUTURE WORK

In this paper, an investigation was held to test different values for the parameters used in the genetic algorithm to find those which can give better results. The system is basically an intrusion detection system which uses detectors generated by genetic algorithm combined with deterministic-crowding niching technique, applied on NSL-KDD IDS test data set under the scope of negative selection theory. Different population sizes against different number of generations with different variation levels were tested. At the stage of rule generated, we use the formal concept analysis to visualize the generated rules, which give more insight analysis on these rules using the visualization. The observations—based on the results shown after running the intrusion detection on the Test Set can be summarized as: (1) Using threshold values 0.2 and 0.4 results in better detection rates (higher true positives). (2) Using higher variation values gave better results when the algorithm was tested against Test Set, which includes unknown attacks that did not exist in the Train Set. (3) Lower population size with more generations gave also the best detection rates when tested on the Test Set. Our future work would be tending to do more investigations on the parameters used in this approach, concerning the features selected for the detection as obviously, real-valued features are not enough to detect attacks, especially there are other features in the data set that are not real-valued but can be used to detect attacks that can not be detected with the features used in the experiment.

## REFERENCES

[1] Varun Chandola, "Anomaly Detection for Symbolic Sequences and Time Series Data", PhD. Dissertation. Computer Science Department, University of Minnesota, September 2009.

[2] Fabio A. Gonzalez and Dipankar Dasgupta, "An Immunity-based Technique to Characterize Intrusions in Computer Networks", Evolutionary Computation, IEEE Trancactions, Vol. 6(3), pp.281-291, 2002.

[3] Fabio A. Gonzalez and Dipankar Dasgupta, "An Immunogenetic Technique to Detect Anomalies in Network Traffic", Proceedings of the Genetic and Evolutionary Computation Conference, GECCO, Morgan Kaufman, pp.1081-1088, 2002.

[4] Tarek S. Sobh and Wael M. Mostafa, “A cooperative immunological approach for detecting network anomaly”, Applied Soft Computing, Elsevier, Vol. 11(1),pp.1275-1283, 2011.

[5] P. Garcia Teodorro, J. Diaz-Verdejo, G. Marcia-Fernandez, E. Vazquez, “Anomaly-based network intrusion detection: Techniques, systems and challenges”, Computers and Security, Elsevier, Vol. 28(1-2), pp.18-28, 2009.

[6] Wei Li, “Using Genetic Algorithm for Network Intrusion Detection”, Proceedings of the United States Department of Energy Cyber Security Grou, Training Conference, Vol. 8, pp. 24-27,2004.

[7] L.N. De Castro and J. Timmis, “Artificial Immune Systems: a new computational intelligence approach”, Springer, Book Chapter, 1st Edition., 2002, XVIII, 380 p.

[8] Dipanker Dasgupta , “Advances in Artificial Immune Systems” ,IEEE Computational Intelligence Magazine, Vol. 1(4), pp.40-49, 2006.

[9] U. Aickelin and D. Dasgupta, “Artificial Immune Systems”, Book Chapter, Search Methodologies: Introductory Tutorials in optimization and decision support techniques, Springer, pp. 375-399, 2003.

[10] Julie Greensmith, Amanda Whitbrook, Uwe Aickelin, “Artificial Immune Systems”, Handbook of Metaheuristics, International Series in Operations Research and Management Science, Springer, Springer US, Vol. 146, pp. 421-448, 2010.

[11] Dipankar Dasgupta, Senhua Yu, Fernando Nino, “Recent Advances in Artificial Immune Systems: Models and Applications”, Applied Soft Computing, Elsevier, Vol. 11(2), pp. 1574-1587, 2011.

[12] Mehdi Kaytoue, Sbastien Duplessis, Sergei O. Kuznetsov and Amedeo Napoli, “Two FCA-Based Methods for Mining Gen Expression Data”, Lecture Notes in Computer Science, Vol. 5548, pp. 251-266, 2009.

[13] Richard Cole, Peter Eklund, Don Walker, “Using Conceptual Scaling In Formal Concept Analysis For Knowledge And Data Discovery In Medical Texts”, Proceedings of the Second Pacific Asian Conference on Knowledge Discovery and Data Mining, pp. 378-379, 1998.

[14] Ole Mengshoel and David E. Goldberg, “The Crowding Approach to Niching in Genetic Algorithms”, Evolutionary Computation, MIT Press Cambridge, Vol. 16(3), pp. 315-354, 2008.

[15] NSL-KDD data set, http://nsl.cs.unb.ca/NSL-KDD/

[16] M. Tavallaee, E. Bagheri, W. Lu, A. A. Ghorbani, “A detailed analysis of the KDD CUP 99 data set”, IEEE Symposium on Computational Intelligence for Security and Defense Applications, CISDA 2009.

[17] WEKA “Waikato Environment for Knowledge Analysis (weka) version 3.5.9”, available on: http://www.cs.waikato.ac.nz/ml/weka/, Junr, 2008.