Anomaly Detection Using Real-Valued Negative Selection†

Fabio A. González (fgonzalz@memphis.edu)
Division of Computer Science, The University of Memphis, Memphis TN 38152 and Departamento de Ingeniería de Sistemas, Universidad Nacional de Colombia.

Dipankar Dasgupta (dasgupta@memphis.edu)
Division of Computer Science, The University of Memphis, Memphis TN 38152

Abstract. This paper describes a real-valued representation for the negative selection algorithm and its applications to anomaly detection. In many anomaly detection applications, only positive (normal) samples are available for training purpose. However, conventional classification algorithms need samples for all classes (e.g. normal and abnormal) during the training phase. This approach uses only normal samples to generate abnormal samples, which are used as input to a classification algorithm. This hybrid approach is compared against an anomaly detection technique that uses self-organizing maps to cluster the normal data sets (samples). Experiments are performed with different data sets and some results are reported.

Keywords: artificial immune systems, anomaly detection, negative selection, matching rule, self-organizing maps

1. Introduction

The anomaly detection problem can be stated as a two-class problem: given an element of the space, classify it as *normal* or *abnormal*. Different terminologies are used in different applications, such as "novelty [9] or surprise [25] detection", "fault detection" [37], and "outlier detection". Accordingly, many approaches have been proposed which include statistical [13], machine learning [29], data mining [30] and immunological inspired techniques [7, 18, 23].

In many anomaly detection applications, however, negative (abnormal) samples are not available at the training stage. For instance, in computer security applications, it is difficult, if not impossible, to have information about all possible attacks. In machine learning approaches, the lack of samples from the abnormal class causes difficulty in applying supervised techniques (e.g. classification). Therefore, the obvious machine learning solution is to use an unsupervised algorithm (e.g. clustering).

† **Draft Version**. Final version appeared in *Genetic Programming and Evolvable Machines, 4(4), pages 383-403, Kluwer Acad. Publ., December 2003.*

Approaches inspired by artificial immune systems have been applied successfully to perform anomaly detection, not only in the area of computer security [8, 21, 23, 26], but also in other fields such as fault tolerant hardware [4], tool breakage detection [9], function optimization [6], etc.

The *negative selection* (**NS**) algorithm [16] has been widely used for change and anomaly detection. The algorithm is inspired by the selection process that takes place inside the thymus. In this process, T-cells that recognize body own cells (self cells) are eliminated; this guarantees that the remaining T-cells will recognize only foreign molecules. D'haeseleer et al. [14] proposed an efficient implementation of the negative selection algorithm known as the *greedy algorithm*. This algorithm is specifically designed for a binary representation of the self/non-self space and the *r-contiguous* matching rule. Though this algorithm has been used in different change/anomaly detection problems, some limitations, however, have prevented it from being applied more extensively:

- Scalability: in order to guarantee good levels of detection, a large number of detectors has to be generated (depending on the size of the self). For some problems, the number of detectors could be unmanageable [27].

- The low-level (binary) detector representation prevents the extraction of meaningful domain knowledge. This makes it difficult to analyze reasons for reporting an anomaly.

- The distinction between the normal and abnormal is considered sharp. This divides the space into two subsets: *self* (the normal) and the *non-self* (abnormal). An element in the space is considered to be abnormal if there exists a detector that matches it. In reality, the normalcy is not a crisp concept. A natural way to characterize the self space is to define a degree of normalcy; this can be accomplished, for instance, by defining the self as a fuzzy set.

- Other immune-inspired algorithms use higher level representation (e.g. real valued vectors). A low level representation, like binary, makes it difficult to integrate the NS algorithm with other immune algorithms.

Some alternatives to r-contiguous matching have been proposed [1, 2, 21]. For example, different matching rules (similarity measures) were reported by Harmer et al. [21]; however, an efficient negative selection algorithm that uses them was not presented. An algorithm that extends the exhaustive detector generation scheme (NS with mutation) was

proposed by Castro and Timmis [12]; this algorithm was compared with other NS implementations by Ayara et al. [1]. Balthrop et al. [2] proposed a different matching rule that subsumes r-contiguous matching, called r-chunks. Some preliminary experiments on a "small data set" suggest that the r-chunk matching rule can improve the accuracy and performance of the NS algorithm.

Gonzalez et al. [20] proposed an approach for the negative selection algorithm that uses a real-valued representation of the self/non-self space. The algorithm is called *real-valued negative selection* (**RNS**). This new algorithm tries to alleviate some of the drawbacks previously mentioned, while using the structure of the higher-level-representation real space to speed up the detector generation process. The algorithm is described in Section 2.

An advantage of using the real-valued representation used is that, for many problems, it is easy to map the generated detectors back to the problem space. This characteristic was used [20] to implement a hybrid approach that combines real-valued negative selection with a conventional classification algorithm to perform anomaly detection tasks. The details of this technique are described in Section 3.

We performed extensive experiments with different data sets. The RNS algorithm is compared with binary NS (**BNS**, NS greedy algorithm [14]) and with an unsupervised technique that builds a model of the normal set through clustering.

A self-organizing map (SOM) [28] is used here as clustering technique, which forms a compact description of the normal space. This compact map is subsequently used to classify new samples as normal or abnormal [17, 24, 34]. A detailed description of this technique is given in Section 4.

2. Real-Valued Negative Selection (RNS)

The NS algorithm [16] is based on the principles of self/non-self discrimination in the immune system. It uses as input, a set of strings that represents the normal data (self set) in order to generate detectors in the non-self space. The negative detectors are chosen by matching them to the self strings: if a detector matches it is discarded, otherwise, it is kept. Some efficient implementations of the algorithm (for binary strings) that run in linear time with the size of self have been proposed [14, 16, 23]. However, the time complexity of these algorithms is exponential on the size of the matching window (the number of bits used to compare two binary strings).

The RNS algorithm was proposed by Gonzalez et al. [20]. Its main feature is that the self/non-self space corresponds to a subset of $\mathbb{R}^n$, specifically $[0,1]^n$.A detector (antibody) is defined by an n-dimensional vector that corresponds to the center and by a real value that represents its radius; therefore, a detector can be seen as a hypersphere in $\mathbb{R}^n$. The matching rule is expressed by the membership function of the detector, which is a function of the detector-antigen Euclidean distance and the radius of the detector (see Equation 1).

The input to the algorithm is a set of self samples represented by n-dimensional points (vectors). The algorithm tries to evolve a complement set of points (called antibodies or detectors) that cover the non-self space. This is accomplished by an iterative process that updates the position of the detector driven by two goals[1]:

- Move the detector away from self points.
- Keep the detectors separated in order to maximize the covering of non-self space.

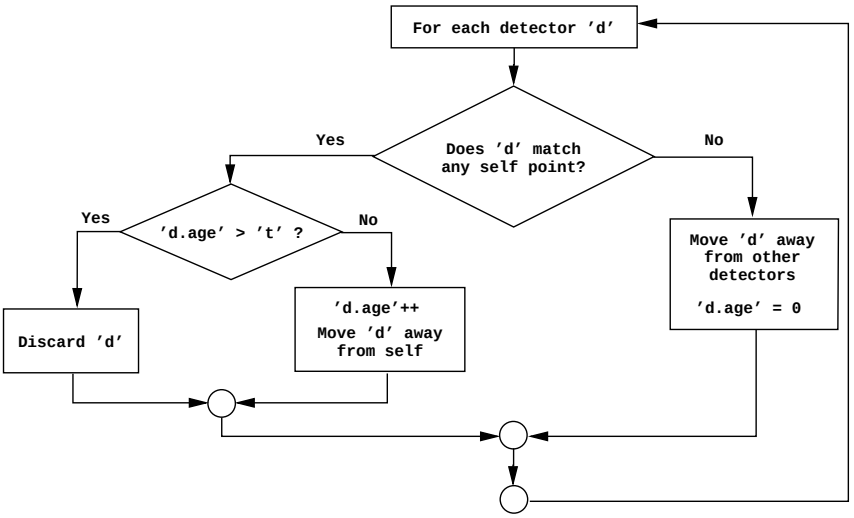


Figure 1. Illustrates an iteration of the real-valued negative selection algorithm.

The logical steps of the algorithm are shown in Figure 1, which are described in a more detailed way in Figure 2.
The parameter r specifies the radius of detection of each detector. Accordingly, for an antigen a to be detected by a detector d, the distance

[1] This approach is similar to the NS greedy algorithm [14], but in a real-valued space.

REAL-VALUED-NEGATIVE-SELECTION(r,η,t,k)

r: radius of detection
η: adaptation rate
t: once a detector reaches this age it will be considered to be mature
k: number of neighbors to take into account

Generate a random population of detectors
While stopping criteria is not satisfied
  For each detector d do
    NearCells = k-nearest neighbors of d in the Self set
    NearCells is ordered with respect to the distance to d
    NearestSelf = median of *NearCells*
    If distance(d, *NearestSelf*) $< r$
     Then:
      $dir = \frac{\sum_{c \in NearCells}(d-c)}{|NearCells|}$
      If age of $d > t$ ▷ detector is old
       Then:
        Replace d by a new random detector
       Else:
        Increase age of d
        $d = d + \eta^* dir$
      EndIf
     Else:
      age of $d = 0$
      $dir = \frac{\sum_{d' \in Detectors} \mu_d(d')(d-d')}{\sum_{d' \in Detectors} \mu_d(d')}$
      $d = d + \eta * dir$
    EndIf
  EndFor
EndWhile

Figure 2. Real-valued negative selection (RNS) algorithm..

between d and a should be at most r. Since we do not want the detectors to match self points, the shortest allowable distance for a good detector to the self set is r. Therefore, the parameter r also specifies the allowed variability of the self space.

In order to determine if a detector d matches a self point, the algorithm calculates the k-nearest neighbors of d in the self set. It then calculates the median distance of these k-neighbors. If this distance is less than r, the detector d is considered to match self. This strategy makes the algorithm more robust to noise and outliers.

The function $\mu_d(x)$ is the matching membership function of the detector d. It indicates the degree of matching between x, an element of the self/non-self space, and d. It is defined as:

$$\mu_d(x) = e^{-\frac{\|d-x\|^2}{2r^2}}, \tag{1}$$

where $\|\cdot\|$ is the Euclidean norm.

Each detector has an assigned age (initialized to zero) that is increased at each iteration, if it is inside the self set. If the detector becomes old, i.e. it reaches the maturity age t and has not been able to move out of the self space, it will be replaced by a new randomly generated detector. The age is reset to zero when the detector is outside of the self space.

The parameter η represents the size of the step used to move the detectors. In order to guarantee that the algorithm will converge to a stable state, it is necessary to decrease this parameter at each iteration in such a way that $\lim_{i\to\infty} \eta_i = 0$. We use the following updating rule

$$\eta_i \leftarrow \eta_o e^{\frac{-i}{\tau}},$$

where η_0 is the initial value of the adaptation rate, and τ is a parameter that controls its decay.

The stopping criterion is based on a pre-specified number of iterations, *num_iter*. This produces a time complexity of $O(num\_iter \cdot num_{ab} \cdot (num_{ab} + |S'|))$, where num_{ab} is the number of detectors and $|S'|$ is the number of self samples.

3. Hybrid Immune System for Anomaly Detection

The NS algorithm has been used mainly to perform negative detection, i.e. the detectors generated by the algorithm are used directly to identify elements in the abnormal (non-self) space. In this section, we present a different use of the NS algorithm to perform anomaly detection that was initially proposed by Gonzalez et al. [20]. This approach uses neither negative nor positive detection; rather, the approach tries to find the boundary (crisp or fuzzy) between normal and abnormal classes. This approach can be useful even if we are not performing distributed anomaly detection or when the self set is small.

The basic idea is to use the RNS algorithm to generate non-self samples. Then, apply a classification algorithm to find a characteristic function of the self (or non-self) space.

Figure 3 illustrates the basic functional blocks of this approach. During the training stage, the input corresponds to the normal samples

(feature vectors), which are used by the RNS algorithm [20] to generate abnormal samples. Subsequently, the normal and abnormal samples are used as input to a supervised algorithm that produces a classifier. This classifier corresponds to the anomaly detection function and is used during the testing phase to classify new samples as normal or abnormal.

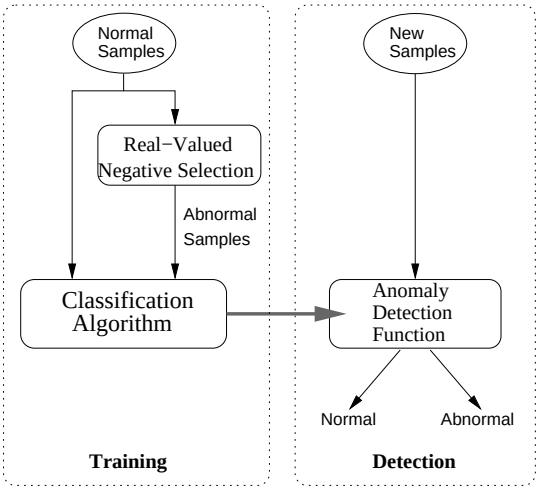


Figure 3. A hybrid immune system for anomaly detection that generates an anomaly characterization function from normal samples.

It is important to highlight that this technique allows the use of a supervised algorithm for a task that traditionally requires an unsupervised method (e.g. clustering). The main advantages of this approach are:

- The classification problem has been studied for a long time. There are different efficient algorithms that have been extensively tested and applied to solve problems in different fields. Our hybrid approach can utilize this algorithms in a efficient way.
- The approach does not require the modification of the classification algorithm. It allows a modular composition that makes easier to use widely available and well tested existing implementations of supervised algorithms.
- The classification problem is closer to the problem of anomaly detection than unsupervised learning problems, like clustering. Clustering methods group the input data based on the principle of maximizing the intraclass similarity and minimizing the interclass similarity. On the other hand, the main objective that drives a

classification algorithm is to improve the accuracy of the classifier, that is, to improve the ability to distinguish between classes. This is clearly more related with the anomaly detection goal of maximizing the detection rate while keeping a low false alarm rate.

– It is possible to use actual abnormal samples, if available, by combining them with the ones generated by the RNS algorithm and inputting them together to the classification algorithm.

The abnormal samples generated by the RNS algorithm can be thought of as *artificial anomalies*. The idea of generating artificial anomalies was independently proposed by Fan et al. [15] and ourselves [20]. Unlike our work, the method proposed by Fan et al. is not related to AIS research, and it does not use or refer to the NS algorithm. The basic idea of Fan's method is to generate artificial anomalies by perturbing real data samples (normal and abnormal). The perturbation is performed by randomly choosing a feature of a given sample and assigning a random value. The generated artificial anomalies are combined with the real data and fed to a learning algorithm. The artificial-anomaly generation method assumes that each feature takes a discrete set of values; hence, the algorithm cannot be directly applied to real-valued data.

There is no specific restriction on the kind of classification algorithm that can be used, i.e., any algorithm that accepts real-valued vectors can work. In this paper, we used a multi-layer perceptron (**MLP**) trained with back-propagation [22]. In the remaining of this paper, we refer this technique as Hybrid Neuro-Immune System (**HNIS**).

4. Anomaly Detection Using Self-Organizing Maps

A self-organizing map (SOM) is a type of neural network that uses competitive learning [22, 28]. A SOM is able to capture the important features contained in the input space and provides a structural representation that preserves a topological structure. The output neurons of a SOM are organized in a one- or two-dimensional lattice. The weight vectors of these neurons represent prototypes of the input data that can be interpreted as the centroids of clusters of similar samples.

In our experiments, we used SOM to cluster the normal samples. After the network is trained, the generated clusters are used to determine if a new sample is normal or abnormal. The basic idea is: if a new sample is 'close' enough to a normal cluster, it is considered normal; otherwise, it is classified as abnormal.

In general, we have a distance function $dist(s, K)$ that measures how close the sample s is to the cluster, K. To determine the abnormality of a new sample, the following function is used:

$$\chi_{abnormal}(s) = \begin{cases} 1 & \text{if } dist(s, Normal) \geq \alpha \\ 0 & \text{otherwise} \end{cases} , \tag{2}$$

where,

$$dist(s, Normal) = \min\{dist(s, K_i) \,|\, K_i \in C\} , \tag{3}$$

and C is the set of clusters (found by the SOM algorithm) that represents the normal sub-space.

If we think the function $dist(s, Normal)$ is a kind of membership function[2] of the abnormal subspace, the function $\chi_{abnormal}(s)$ corresponds to the crisp version of it. In this case, the value α represents a threshold that defines the boundary between the normal and abnormal classes.

In order to determine a good distance measure $dist(s, K)$, we tested three options (in all the cases w_K, neuron weights, represents the centroid of the cluster K):

- **Euclidean distance**. This is the natural (or naive) choice since the SOM algorithm uses it to determine if a sample belongs to a given cluster:
$$dist(s, K) = \|s - w_K\| \tag{4}$$
- **Normalized distance**. The idea is to take into account the size of the cluster. Some clusters can be very sparse and others can have all the elements concentrated around the centroid. A measure of the size is the standard deviation. So, the standard deviation of the distance to the centroid of all the elements in a cluster (σ_K) is calculated and it is used to normalize the distance:
$$dist(s, K) = \frac{\|s - w_K\|}{\sigma_K} \tag{5}$$
- D_∞ **Minkowsky distance**. The Euclidean distance gives the same importance to all the features. So, it is possible that a sample with a non-negligible deviation in one feature will be considered as having the same overall deviation as a pattern with small deviations on many features. The D_∞ distance only takes into account the maximum of the differences for all the features:
$$dist(s, K) = \max\{|s_i - w_{K_i}| \,\text{for}\, i = 1, \ldots, n\} \tag{6}$$

[2] Strictly speaking, this is not a membership function since it is not bounded. However, we can apply, for instance, a sigmoid function to make it bounded.

5. Experiments and Results

We used four different data sets to test the ability of the proposed technique, HNIS, to generate a good anomaly detection function when it is trained using only normal samples. Each data set is divided in two subsets: the training data set, which contains only normal samples, and the test data set, containing a mixture of normal and abnormal samples.

In order to compare the performance, we also apply BNS (greedy algorithm) and the SOM anomaly detection technique (explained in Section 4) to the same data sets. These techniques are compared in terms of classification accuracy. The idea is to calculate the number of true positives (TP, anomalous elements identified as anomalous), true negatives (TN, normal elements identified as normal), false positives (FP, normal elements identified as anomalous) and false negatives (FN, anomalous elements identified as normal). These values are, however, used to calculate two measures of effectiveness:

$$\text{Detection rate} = \frac{TP}{TP + FN}$$

$$\text{False alarm rate} = \frac{FP}{TN + FP}$$

In general, we want a very high detection rate with a very low false alarm rate. However, there is a trade-off between these two measures. This trade-off can be shown using ROC (receiver operating characteristics) curves [35]. The sensitivity of the system is controlled by a threshold that determines when a new sample is normal or abnormal. By varying this threshold, we can obtain different values for the detection and false alarm rates which are used to plot ROC curves. In the case of HNIS and SOM techniques, whose output is a continuous value, the threshold value is considered between 0 and 1.

The output of the BNS technique is not continuous, it is just 0 (normal) or 1 (abnormal); so, it does not make sense to apply a threshold to this output. It is possible to use the parameter r, which define the size of the matching window, as a threshold that we can vary to produce different points of the ROC curve. However, we have to be careful when interpreting the results, since, unlike the other two methods (HNIS and SOM), each point of the ROC curve in BNS will correspond to a different set of detectors.

5.1. Mackey-Glass time series experiments

5.1.1. *Data set and preprocessing*

The Mackey-Glass series [31] has been used as a test set for different anomaly detection approaches [5, 9, 25]. Although it is generated by a deterministic differential equation, it exhibits a chaotic behavior that makes its prediction difficult.

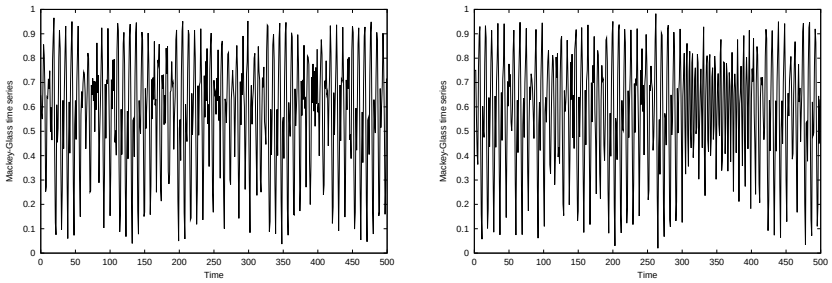
The differential equation that defines the series is the following:

$$\frac{dx}{dt} = \frac{ax(t-\tau)}{1+x^c(t-\tau)} - bx(t) \tag{7}$$

The parameter τ controls the complexity of the series dynamics, ranging from periodic to chaotic behavior.

In order to generate the training and testing data sets, Equation 7 is solved numerically using the fourth-order Runge-Kutta method with an integration step of 0.02, a sampling rate of 12, and an initial value vector with all its elements equal to 1.1. The parameter values used for the equation are: $a = 0.2$, $b = 0.1$, and $c = 10$, which are the general choice in the literature [9, 5].

The normal samples were produced from a time series with 500 elements generated using $\tau = 30$ and discarding the first 1000 samples to eliminate the initial value effect. The resulting time series is shown in Figure 4(a).



(a) (b)

Figure 4. Mackey-Glass time series: (a) normal, using $\tau = 30$, (b) with an anomaly, $\tau = 17$ from 300 to 400.

The test data (Figure 4(b)) is generated as before using $\tau = 30$, but starting with different initial conditions. An abnormality is introduced between time 300 to 400 by changing the parameter τ to 17. It is important to note that this experimental setting is different from the one used by Dasgupta and Forrest [9]. In that work, the anomalous time series is identical to the normal one, with the exception of the portion between 1000 and 1500. In our case, the two series are completely

different since they were generated using different initial conditions. This makes the problem more challenging for the anomaly detection algorithm, since it has to be able to learn the structure of the normal set.

The features are extracted using a sliding overlapping window of size n. If the time series has the values: $x_1, x_2, ..., x_m$, the feature set generated from it will be the following:

$$\begin{array}{llll} (x_1, & x_2, & ... & x_n) \\ (x_2, & x_3, & ... & x_{n+1}) \\ \vdots & \vdots & \vdots & \vdots \\ (x_{m-n+1} & x_{m-n+2} & ... & x_m) \end{array}$$

So, from a time series with m elements and using a sliding window of size n, we can generate $(m\text{-}n\text{+}1)$ samples. In our experiments, we used $n = 4$. All the vector components are normalized to the interval [0,1].

5.1.2. *Experimental settings*

For the HNIS technique, the RNS algorithm was run using as input the training data to generate 400 detectors. The parameter values for the algorithm were: $r = 0.1$, $\eta = 1$, $t = 5$, and $k = 1$. The number of iterations was set to 400. The MLP used had 4 inputs and 1 output. Three different architectures were tested with 6, 12, and 16 hidden neurons. The parameters of the back-propagation algorithm were: learning rate 0.05, momentum 0.9, and number of iterations 4000.

For the BNS algorithm, the data was converted to binary strings assigning 5 bits to each feature and using binary and gray coding. This produced binary strings of length 20. The matching threshold, r, was varied from 6 to 12. The greedy algorithm was run setting the failure probability to 0.

Three different SOM topologies were used: 4 input nodes with an output layer of 4×3, 4×6 and 6×6 neurons, respectively. The weights were initialized using random vectors. The SOM training algorithm was run for 1000 iterations using a Gaussian neighborhood. The initial and final learning rate were 0.1 and 0.005 respectively. The initial σ value was 5 and the final was 0.2.

5.1.3. *Results*

Figure 5 shows a typical output of three techniques when applied to the testing set. An output value close to "0" means normalcy. Each figure corresponds to the best result found by each method. Despite the fact that in all cases the output shows an increased activity in

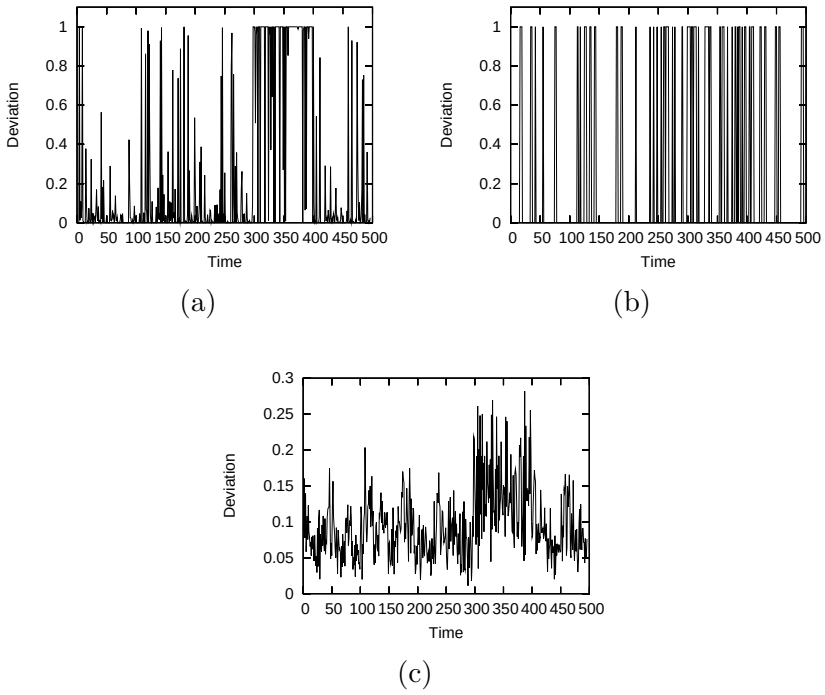


Figure 5. Output value produced by the anomaly function when applied to the Mackey-Glass testing set. (a) HNIS (12 hidden neurons); (b) BNS ($r = 8$, Gray coding); (C) SOM (6×6, D_∞ distance).

the abnormal region, there are peaks in the normal region that do not allow to establish a clear boundary between normal and abnormal. In order to smooth the anomaly detection function, a moving average filter was applied. Accordingly, the new output $\widehat{O_t}$ is calculated from the old output O_t using the following formula:

$$\widehat{O_t} = \frac{\sum_{i=1}^{s} O_{t-i}}{s}, \tag{8}$$

where s is the smoothing factor and indicates the size of the averaging window. The filter was applied to the output produced by each technique. Different values of s were tested (5, 10 and 15), choosing the value that produced the best result for each individual technique. Figure 6 shows the smoothed versions of the outputs in Figure 5.

The following subsections shows more details of the results produced by each technique.

5.1.3.1. *BNS results* The number of detectors generated by the NS greedy algorithm are summarized in Table I. These values coincide with

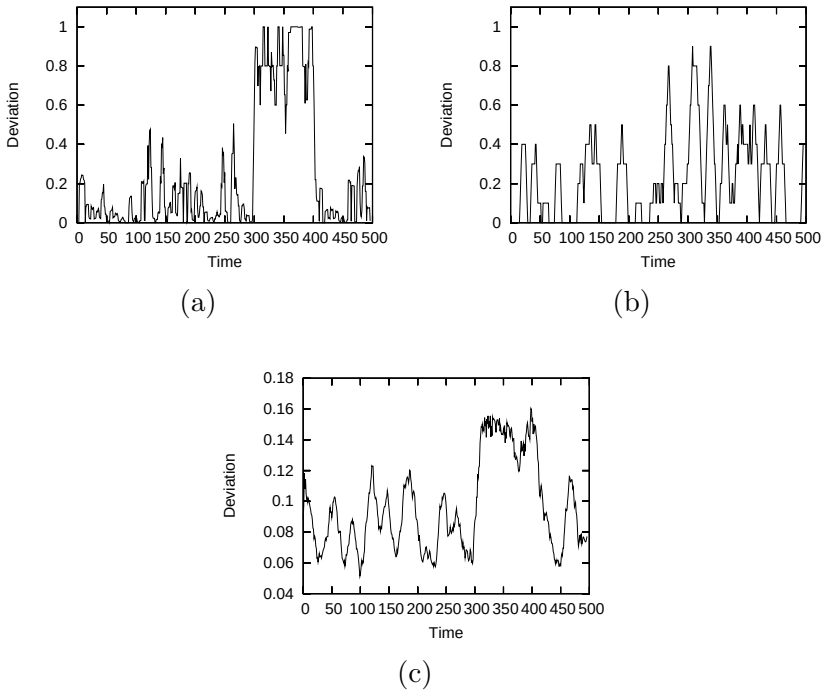


Figure 6. Output value, smoothed using Equation 8, produced by the anomaly function when applied to the Mackey-Glass testing set. (a) HNIS (12 hidden neurons, $s = 5$); (b) BNS ($r = 8$, Gray coding, $s = 10$); (C) SOM (6×6, , D_∞ distance, $s = 10$).

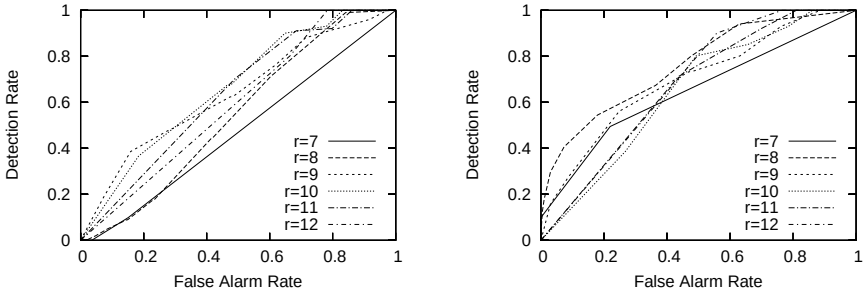
the values predicted by the theoretical analysis described by D'haeseleer et al. [14].

The performance of the different set of detectors is shown in the ROC curves in Figure 7. It is important to note that it is possible to generate these ROC curves for each detector because of the smoothing process. This generates a continuous anomaly function that takes values between 0 and 1; so, it makes sense to use a threshold to decide when a given sample is normal or abnormal.

The results using Gray coding are in general better than the results produced with Binary coding. This is explained by the fact that Gray coding is more compatible with the kind of matching rule used by the BNS algorithm, r-contiguous matching. This is a fact that has been addressed by Dasgupta and Majumdar [11]. The best result is produced with a set of detectors generated using $r = 8$. An increase in r does not improve the performance, as it is shown by the ROC curves for $r = 9$ to $r = 12$, which are bound by the ROC curve generated with $r = 8$.

Table I. Number of detectors produced by BNS (greedy) algorithm when applied to the Mackey-Glass training set.

	Number of detectors	
r	**Binary coding**	**Gray coding**
6	0	0
7	13	19
8	90	79
9	300	301
10	736	750
11	1683	1705
12	3691	3709



(a) Binary encoding (b) Gray encoding

Figure 7. ROC curves for the BNS algorithm applied to the Mackey-Glass test data set.

5.1.3.2. *SOM results* As it was discussed in section 4, three different distance measures were proposed to calculate the anomaly detection function defined in Equation 2. Figure 8(a) shows the ROC curves corresponding to these distance measures. D_∞ Minkowsky distance (Equation 6) shows a slight advantage over other distance measures. Figure 8(b) shows ROC curves for different topologies of the SOM network. A higher number of neurons produces a most accurate classification; however, the difference between the curves is not big; this suggests that a further increase in the network complexity may not improve the accuracy.

5.1.3.3. *HNIS results* Figure 9 shows the ROC curves corresponding to different MLP topologies. The figure shows that an increase from

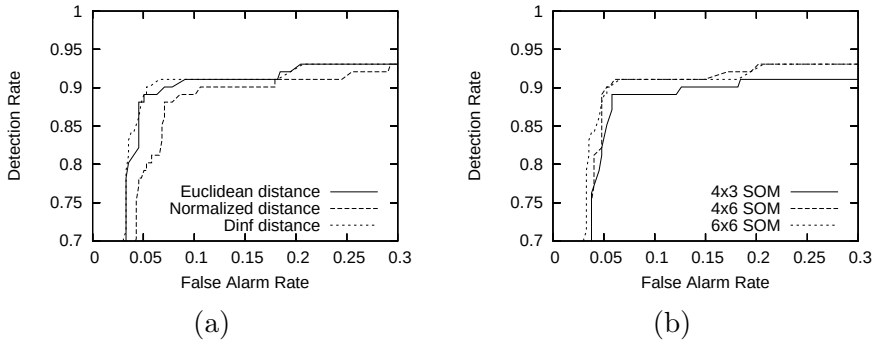


Figure 8. ROC curves for SOM anomaly detection applied to Mackey-Glass test data set. (a) different distance measures using 6×6 topology; (b) different topologies using D_∞ distance.

6 to 12 neurons improves the classification accuracy of the system. Accordingly, 12 neurons seem to be enough, since an increase to 16 does not produce any significant improvement in accuracy.

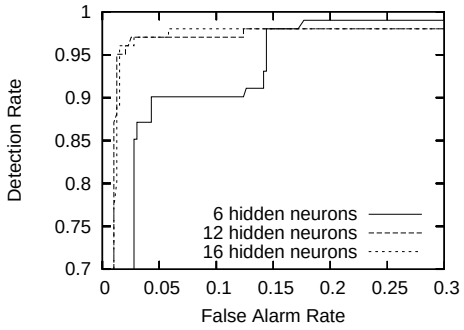


Figure 9. ROC curves for HNIS anomaly detection applied to Mackey-Glass test data set for different MLP topologies: 6, 12, and 16 hidden neurons.

5.1.4. *Results comparison and discussion*

The best performing configurations from each approach are compared in Figure 10. The configurations are: HNIS, 12 hidden neurons; BNS, Gray coding and $r = 8$; and SOM, 6×6 output layer and D_∞ distance. Clearly, HNIS has a better performance than other two methods. This shows that, at least for this specific data set, the combination of RNS with a MLP is able to capture the structure of the normal space, producing an anomaly detection function that can discriminate the normal and the abnormal new samples.

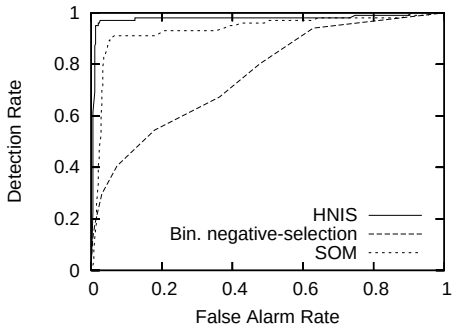


Figure 10. Best ROC curves produced by each method for the Mackey-Glass test data set.

In contrast to the results reported by Dasgupta and Forrest [10], the performance of the BNS algorithm is very poor. This may be because of the experimental settings used in our current work; the normal samples in the test data set are different from to those presented during training. This indicates that the anomaly detection algorithm should be able to generalize the structure of the normal set based on a limited subset of samples. Our hypothesis is that the binary (low-level) representation along with the r-contiguous matching rule (used by BNS) may not capture the high-level structure of the problem space.

5.2. Network traffic data experiments

5.2.1. *Data sets*

5.2.1.1. *MIT-Darpa 98* This data set is a version of the 1998 DARPA intrusion detection evaluation data set prepared and managed by MIT Lincoln Labs [32]. The data set was generated by processing the original tcpdump data to extract 42 attributes (33 of them numerical) that characterize the network traffic. This set was used in the *KDD Cup 99* competition and is available at at the University of Irvine Machine Learning repository[3] [33]. Even though the data set corresponds to a 10% of the original data, its size is still considerably big (492,021 records).

We generated a reduced version of the 10% data set including only the numerical attributes. Therefore, the reduced 10% data set is composed by 33 attributes. The attributes were normalized between 0 and 1 using the maximum and minimum values found. Of the normal samples, 80% were picked randomly and used as training data set, while the remaining 20% was used along with the abnormal samples as a testing

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

set. Five fuzzy sets were defined for the 33 attributes. One percent of the normal data set (randomly generated) was used as a training data set.

5.2.1.2. *MIT-Darpa 99* This data set, was directly obtained from the MIT-Lincoln Lab [32], and corresponds to the 1999 DARPA intrusion detection system evaluation program. 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, 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.

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. Each set contains approximately 5000 records.

5.2.2. *Experimental settings*
The experimental settings for all the techniques are the same as the ones described in section 5.1.2. The only differences are: for the MIT-Darpa 98 data set, the HNIS used 1000 detectors instead of 400, and for the MIT-Darpa 99 data set, three different MLP topologies were tested with 5, 9, and 18 hidden neurons, respectively.

5.2.3. *Results comparison and discussion (MIT-Darpa 98)*
The BNS algorithm was not able to generate a good set of detectors. We ran it for different values of r ranging from 6 to 12. The algorithm did not produce detectors for values of r less or equal to 8; however, for $r = 9$ the algorithm produced more than 2×10^8 detectors before it had to be manually stopped. This happened even with a failure probability as high as 0.5. Our hypothesis is that the high dimensionality of the space along with the small variability of the normal set makes it very difficult to cover the non-self space using the NS greedy algorithm. This result is similar to the one reported by Kim and Bentley [27].

Figure 11 shows the best ROC curves produced by the HNIS and SOM anomaly detection techniques. The configurations are: HNIS, 12

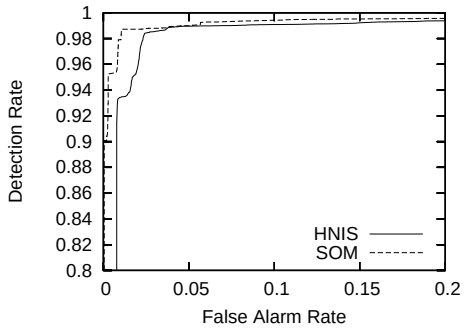


Figure 11. Best ROC curves produced by HNIS and SOM methods for the MIT-Darpa 98 test data set.

hidden neurons and SOM, 4×6 using D_∞ distance. The performance of the two techniques was similar with a slight advantage of the SOM technique. The performance of the HNIS is remarkable, in a problem that seems to be very difficult for a technique that generates non-self detectors in such a high dimensional space. It is clear that 1000 detectors are not enough to cover this space; however, the experiments showed that they were enough to train a classifier (MLP) that could effectively discriminate between normal and abnormal samples in the testing set.

5.2.4. *Results comparison and discussion (MIT-Darpa 99)*
Figure 12 shows the best ROC curves produced by the three techniques. The configurations are: HNIS, 5 hidden neurons; SOM, 4×6 output layer using D_∞ distance; and BNS, $r = 6$ with binary or Gray coding. The SOM method is clearly better that the other two methods. However, the other two methods also produced good results that have a detection rate over 93% with a false alarm rates as small as 1%. The HNIS method can reach a detection rate as accurate as the one produced by SOM (98%), but only if the false alarm rate is increased to 13%. Notice that this trade-off cannot be applied to the BNS. However, the BNS produces a very good detection rate (95%) with a very small false alarm rate.

5.3. Wisconsin breast cancer experiments

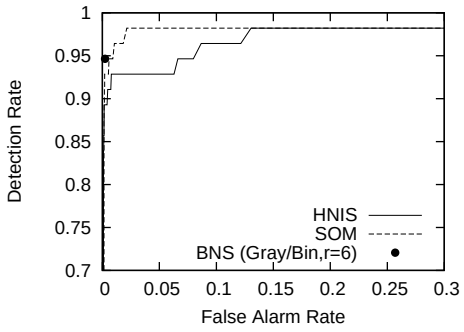


Figure 12. Best ROC curves produced by each method for Darpa 99 test data set.

5.3.1. *Data set*
This data set correspond to a breast cancer data set created at the University of Wisconsin Hospitals [36]. This particular data set was obtained from the University of Irvine Machine Learning repository[4]. Each data record is conformed by ten numerical attributes and the label (benign or malign). The data is composed by 699 records, but 16 of them have missing values. (we did not use these records.) The data was normalized to fit the interval [0,1] , and we partitioned it in two sets, training and testing. The training set contains 271 benign records. The testing set is composed of 412 mixed benign and malign records.

5.3.2. *Experimental settings*
The experimental settings for all three techniques are the same as the ones described in Section 5.1.2.

5.3.3. *Results comparison and discussion*
The best ROC curves produced by each method are shown in Figure 13. These curves are produced by the following configurations: HNIS, 18 hidden neurons and SOM, 4×6 output layer using Euclidean distance. In the case of BNS, there are three good configurations: $r = 7$ with Gray coding, $r = 8$ with Gray coding, and $r = 4$ with binary coding. It is important to note that the points in the ROC diagram for the BNS method are generated by three different runs of the algorithm, whereas the points for the other two methods correspond to only one run in each case. All of the methods are able to produce high detection rates. The HNIS method has a slight advantage over the SOM method,

[4] Original database at `ftp://ftp.ics.uci.edu/pub/machine-learning-databases/breast-cancer-wisconsin`.

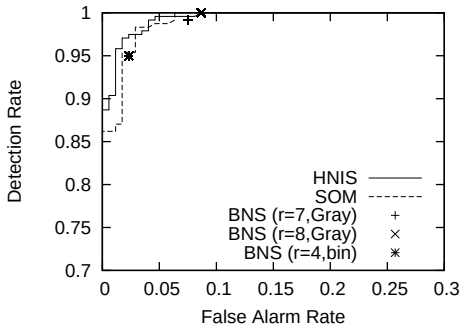


Figure 13. Best ROC curves produced by each method for Wisconsin breast cancer test data set.

mainly for small false alarm rates. For false alarm rates higher than 7%, the performance of all the methods is similar.

6. Conclusions

In this paper, we presented a hybrid anomaly detection technique (HNIS) that combines an immune inspired algorithm, real-valued negative selection (which is also presented), and a conventional classification algorithm. This method does not use positive or negative detection. Rather, it tries to find a boundary between normal and abnormal classes.

The hybrid method is compared against binary negative selection (BNS), using the greedy algorithm with r-contiguous matching [14], and an anomaly detection technique based on self-organizing maps (SOM).

The proposed approach (HNIS) produced good results in all four data sets. BNS performed well in two of the experiments; however, it failed to produce acceptable results in two other cases. The MIT-Darpa 98 data set is one of the data sets where BNS failed. This is consistent with the results reported by Kim and Bentley [27]; these results were used by them to support the claim that negative selection algorithm suffers from "severe scaling problems". However, our work shows that the problem is not with the negative selection algorithm itself, rather the kind of representation (binary) and matching rule (r-contiguous) that were used. This was also suggested by Balthrop et al. [3].

Another important characteristic of the proposed approach is that it can learn the structure of the self set using only a subset of normal samples. In some applications, mainly in change detection, it is assumed that the self set is complete; however, in many real anomaly detection

applications, this is not the case. Hence, an anomaly detection algorithm must be able to produce a good approximation of the structure of the self/non-self space, even if a portion of the self set is available during training. The experiments with the Mackey-Glass data set (Section 5.1) are a good example of such problem. The BNS algorithm was not able to make a good generalization of the self set for this data set, resulting in a poor anomaly detection performance. The main reason is that the low-level (binary) representation and the matching rule (r-contiguous) used by the BNS algorithm do not represent appropriately the affinity relationship at the problem space [19].

The results produced by the SOM method are similar to the ones produced by the HNIS. It is not possible to conclude that one technique outperformed the other, since the differences on performance are not statistically significant. These results are encouraging since they suggest that the proposed hybrid approach can produce results that are competitive with those produced by positive detection while providing some additional characteristics (such as the possibility of using a classification algorithm when only normal samples are available).

Finally, the use of a more expressive representation for the detectors allows the combination of negative selection with other learning methods. Our previous work demonstrated the feasibility of combining the negative selection algorithm with a classification algorithm (a MLP trained with back-propagation). A very interesting experiment would be to combine it with other immune inspired techniques, like those based on immune network theory. This would open the doors for the construction of an unified artificial system that combines different types of immune mechanisms.

7. 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. Ayara, M., J. Timmis, L. de Lemos, R. de Castro, and R. Duncan: 2002, 'Negative selection: How to generate detectors'. In: J. Timmis and P. J. Bentley (eds.): *Proceedings of the 1st International Conference on Artificial Immune Systems (ICARIS)*. Canterbury, UK, pp. 89–98.
2. Balthrop, J., F. Esponda, S. Forrest, and M. Glickman: 2002a, 'Coverage And Generalization In An Artificial Immune System'. In: W. B. Langdon, E. Cantú-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G.

Rudolph, J. Wegener, L. Bull, M. A. Potter, A. C. Schultz, J. F. Miller, E. Burke, and N. Jonoska (eds.): *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*. San Francisco, CA, pp. 3–10.

3. Balthrop, J., S. Forrest, and M. R. Glickman: 2002b, 'Revisting LISYS: Parameters and Normal Behavior'. In: D. B. Fogel, M. A. El-Sharkawi, X. Yao, G. Greenwood, H. Iba, P. Marrow, and M. Shackleton (eds.): *Proceedings of the 2002 Congress on Evolutionary Computation CEC2002*. USA, pp. 1045–1050.
4. Bradley, D. and A. Tyrrell: 2002, 'Immunotronics: Novel Finite-State-Machine Architectures with Built-In Self-Test Using Self-Nonself Differentiation'. *IEEE Transactions on Evolutionary Computation* **6**(3), 227–238.
5. Caudell, T. and D. Newman: 1993, 'An adaptive resonance architecture to define normality and detect novelties in time series and databases'. In: *IEEE World Congress on Neural Networks*. Portland, OR, pp. 166–176.
6. Coello Coello, C. A. and N. Cruz Cortés: 2002, 'An Approach to Solve Multiobjective Optimization Problems Based on an Artificial Immune System'. In: J. Timmis and P. J. Bentley (eds.): *First International Conference on Artificial Immune Systems (ICARIS)*. Canterbury,UK, pp. 212–221.
7. Dagupta, D. and F. González: 2002, 'An Immunity-Based Technique to Characterize Intrusions in Computer Networks'. *IEEE Transactions on Evolutionary Computation* **6**(3), 281–291.
8. Dasgupta, D.: 1999, *Artificial immune systems and their applications*. New York: Springer-Verlag.
9. Dasgupta, D. and S. Forrest: 1996, 'Novelty detection in time series data using ideas from immunology'. In: J. F. C. Harris (ed.): *Proceedings of the 5th International Conference on Intelligent Systems*. Cary, NC, pp. 82–87.
10. Dasgupta, D. and S. Forrest: 1999, 'An anomaly detection algorithm inspired by the immune system'. In: D. Dasgupta (ed.): *Artificial immune systems and their applications,*. New York: Springer-Verlag, pp. 262–277.
11. Dasgupta, D. and N. S. Majumdar: 2002, 'Anomaly Detection in Multidimensional Data using Negative Selection Algorithm'. In: D. B. Fogel, M. A. El-Sharkawi, X. Yao, G. Greenwood, H. Iba, P. Marrow, and M. Shackleton (eds.): *Proceedings of the 2002 Congress on Evolutionary Computation (CEC2002)*. USA, pp. 1039–1044.
12. de Castro, L. N. and J. Timmis: 2002, *Artificial Immune Systems: A New Computational Approach*. London, UK: Springer-Verlag.
13. Denning, D. E.: 1987, 'An intrusion-detection model'. *IEEE Transactions on Software Engineering* **13**(2), 222–232.
14. D'haeseleer, P., S. Forrest, and P. Helman: 1996, 'An immunological approach to change detection: algorithms, analysis and implications'. In: J. McHugh and G. Dinolt (eds.): *Proceedings of the 1996 IEEE Symposium on Computer Security and Privacy*. USA, pp. 110–119.
15. Fan, W., W. Lee, M. Miller, S. Stolfo, and P. Chan: 2001, 'Using artificial anomalies to detect unknown and known network intrusions'. In: N. Cercone, T. Y. Lin, and X. Wu (eds.): *Proceedings of the 1st IEEE International conference on Data Mining*. USA, pp. 123–130.
16. Forrest, S., A. Perelson, L. Allen, and R. Cherukuri: 1994, 'Self-nonself discrimination in a computer'. In: *Proceedings IEEE Symposium on Research in Security and Privacy*. Los Alamitos, CA, pp. 202–212.
17. Fox, K., R. Henning, J. Reed, and R. Simonian: 1990, 'A neural network approach towards intrusion detection'. In: *Proc. 13th NIST-NCSC national computer security conference*. Washington, DC, pp. 125–134.

18. González, F. and D. Dasgupta: 2002, 'An imunogenetic technique to detect anomalies in network traffic'. In: W. B. Langdon, E. Cantú-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. A. Potter, A. C. Schultz, J. F. Miller, E. Burke, and N. Jonoska (eds.): *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*. San Francisco, CA, pp. 1081–1088.
19. Gonzalez, F., D. Dasgupta, and J. Gomez: 2003, 'The Effect of Binary matching Rules in Negative Selection'. In: *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*.
20. González, F., D. Dasgupta, and R. Kozma: 2002, 'Combining negative selection and classification techniques for anomaly detection'. In: D. B. Fogel, M. A. El-Sharkawi, X. Yao, G. Greenwood, H. Iba, P. Marrow, and M. Shackleton (eds.): *Proceedings of the 2002 Congress on Evolutionary Computation CEC2002*. USA, pp. 705–710.
21. Harmer, P., G. Williams, P.D.and Gnusch, and G. Lamont: 2002, 'An Artificial Immune System Architecture for Computer Security Applications'. *IEEE Transactions on Evolutionary Computation* **6**(3), 252–280.
22. Haykin, S.: 1994, *Neural networks : a comprehensive foundation*. New York: Macmillan.
23. Hofmeyr, S. and S. Forrest: 2000, 'Architecture for an Artificial Immune System'. *Evolutionary Computation* **8**(4), 443–473.
24. Hsu, W., L. Auvil, W. Pottenger, D. Tcheng, and M. Welge: 1999, 'Self-organizing systems for knowledge discovery in databases'. In: *In proceedings of the international joint conference on neural networks IJCNN-99*. USA.
25. Keogh, E., S. Lonardi, and B. Chiu: 2002, 'Finding surprising patterns in a time series database in linear time and space'. In: O. R. Zaïane, R. Goebel, D. Hand, D. Keim, and R. Ng (eds.): *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '02)*. USA, pp. 550–556.
26. Kephart, J. O.: 1994, 'A Biologically Inspired Immune System for Computers'. In: R. A. Brooks and P. Maes (eds.): *Proceedings of the 4th International Workshop on the Synthesis and Simulation of Living Systems $ArtificialLife IV$*. Cambridge, MA, USA, pp. 130–139.
27. Kim, J. and P. Bentley: 2001, 'An Evaluation of Negative Selection in an Artificial Immune System for Network Intrusion Detection'. In: L. Spector, E. D. Goodman, A. Wu, W. B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, and E. Burke (eds.): *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*. San Francisco, CA, pp. 1330–1337.
28. Kohonen, T.: 1995, *Self-Organizing Maps*, Vol. 30 of *Springer Series in Information Sciences*. Berlin, Heidelberg: Springer. (Second Extended Edition 1997).
29. Lane, T.: 2000, 'Machine learning techniques for the computer security'. Ph.D. thesis, Purdue University.
30. Lee, W. and S. Stolfo: 1998, 'Data mining approaches for intrusion detection'. In: *Proceedings of the 7th USENIX security symposium*. Berkeley, CA, pp. 79–94.
31. Mackey, M. and L. Glass: 1977, 'Oscillation and chaos in physiological control systems'. *Science* **197**, 287–289.
32. MIT: 1999, '1999 Darpa intrusion detection evaluation'. MIT Lincoln Labs.
33. Murphy, P. and D. Aha: 1992, 'UCI Repository of machine learning databases'.

34. Portnoy, L., E. Eskin, and S. Stolfo: 2001, 'Intrusion detection with unlabeled data using clustering'. In: *Proceedings of ACM CCS Workshop on Data Mining Applied to Security*. USA.
35. Provost, F., T. Fawcett, and R. Kohavi: 1998, 'The case against accuracy estimation for comparing induction algorithms'. In: J. Shavlik (ed.): *Proceedings of 15th International Conference on Machine Learning*. San Francisco, CA, pp. 445–453.
36. Wolberg, W. H. and O. Mangasarian: 1990, 'Multisurface method of pattern separation for medical diagnosis applied to breast cytology'. *Proceedings of the National Academy of Sciences, U.S.A.* **87**, 9193–9196.
37. Yoshikiyo, T.: 2001, 'Fault detection by mining association rules from housekeeping data'. In: *proceedings of international symposium on artificial intelligence, robotics and automation in space (i-sairas 2001)*. Montreal, Canada.