A Randomized Real-Valued Negative Selection Algorithm

Fabio González[1,2], Dipankar Dasgupta[2], and Luis Fernando Niño[1]

[1] Depto. de Ing. de Sistemas, Universidad Nacional de Colombia, Bogotá, Colombia
[2] Division of Computer Science, The University of Memphis , Memphis TN 38152
{fgonzalz,dasgupta}@memphis.edu, lfnino@ing.unal.edu.co

Abstract. This paper presents a real-valued negative selection algorithm with good mathematical foundation that solves some of the drawbacks of our previous approach [11]. Specifically, it can produce a good estimate of the optimal number of detectors needed to cover the non-self space, and the maximization of the non-self coverage is done through an optimization algorithm with proven convergence properties. The proposed method is a randomized algorithm based on Monte Carlo methods. Experiments are performed to validate the assumptions made while designing the algorithm and to evaluate its performance.[3]

1 Introduction

The negative selection (**NS**) algorithm is one of the most widely used techniques in the field of artificial immune systems. It is primarily used to detect changes in data/behavior patterns by generating detectors in the complementary space (given normal samples). In the original version of the NS algorithm [8], the detectors are used directly to classify new data as self (normal) or non-self (abnormal). Subsequent works have shown the feasibility of combining the NS algorithm with classification algorithms [11]; in this case, the generated detectors are used by the classification algorithm to learn high-level anomaly detection functions. Regardless of how the detectors are used, a good coverage of the non-self space is important for the anomaly detection process.

This paper focuses on the problem of efficient generation of detectors when a real-valued representation of the self/non-self space is used. Other important issues concerning the NS algorithm are discussed elsewhere (positive vs negative detection [6,9], representation and matching rules [10,12], applications[3]), and thus they are not considered in this paper.

González et al. [11] proposed a Real-Valued Negative Selection (**RNS**) algorithm based on heuristics that try to distribute the detectors in the non-self space in order to maximize the coverage. This algorithm uses a real-valued representation for the self/non-self space that differs from the binary representation used in original negative selection algorithms [5,8]. This higher-level representation provides some advantages such as increased expressiveness, the possibility of extracting high-level knowledge from the generated detectors, and, in some cases, improved scalability [9,11]. However,

[3] In *Proceedings of the 2nd International Conference on Artificial Immune Systems*, pp 261–272, Edinburgh, UK, September 2003.

this algorithm lacks the theoretical support of the binary negative selection algorithm [5,6]. The main difficulties due to the lack of theoretical support include:

– The number of detectors needed to cover the non-self space, as well as the radius of each detector, are not known in advance; hence, it is necessary to determine them by a trial-and-error procedure.
– There is no guarantee that the algorithm will converge to an optimal or close-to-optimal space coverage with minimum overlap.

This paper proposes a randomized RNS algorithm (**RRNS**) based on some mathematical models, which can provide specific criteria to setup the algorithmic parameters and to assess the expected performance. The proposed algorithm is based on two main ideas:

– To estimate the volume of the self space, which, by complementarity, is also an approximation of the volume of the non-self space. Using this volume, it is possible to calculate how many hyper-spherical detectors (of a given radius) are needed to cover the non-self space.
– To use a well known optimization algorithm, simulated annealing [13,16], to find a good distribution of the detectors that maximizes the coverage of the non-self space.

The algorithm is called *randomized* because it is based on an important class of randomized algorithms known as Monte Carlo methods [2,7,14]. Specifically, it uses *Monte Carlo integration* to estimate the volume of the self (and non-self) space and *simulated annealing* [13,16] to optimize the distribution of detectors in the non-self space.

It is to be noted that this is not the first time that simulated annealing has been used in an artificial immune algorithm. De Castro and Von Zuben [4] proposed a technique to initialize feed-forward neural network weights, where the basic idea is to represent the network weights by detectors which correspond to *n*-dimensional real-valued vectors. The detectors are dispersed in the space by maximizing an energy function that takes into account the inverse of the inter-detector affinity. De Castros's approach is substantially different from the one proposed here; his approach does not use the concept of self/non-self distinction, and its main goal is producing diversity instead of performing anomaly detection.

2 Randomized Real-Valued Negative Selection Algorithm (RRNS)

Similar to the RNS algorithm, the objective of this algorithm is to generate a set of hyper-spherical detectors that cover the non-self space. The algorithm primarily consists of two steps: first, it generates an initial set of detectors; second, it optimizes the distribution of this set to maximize the non-self coverage. The input to the algorithm is a set of samples from the self set, S'; the allowed variability in the self set, r_{self}; the detector radius, r_{ab}; and a set of parameters, Π. The global structure of the algorithm is shown in Figure 1.

Accordingly, the algorithm is implemented with two main functions: CALCULATE-INIT-DETECTOR-SET (described in Section 2.1), which estimates the volume of the

```
RR-NEGATIVE-SELECTION($S'$, $r_{self}$, $r_{ab}$, $\Pi$)
    $S'$ : set of self samples
    $r_{self}$ : self variability threshold
    $r_{ab}$ : detector radius
    $\Pi$ : additional parameters
1: $D \leftarrow$CALCULATE-INIT-DETECTOR-SET($S'$, $r_{self}$, $r_{ab}$)
2: $D' \leftarrow$OPTIMIZE-DETECTOR-DISTRIBUTION($D$, $r_{ab}$, $S'$, $r_{self}$)
3: Return $D'$
```

Fig. 1. Randomized real-valued negative selection (RRNS) algorithm.

non-self space in order to produce a good initial set of detectors, and OPTIMIZE-DETECTOR-DISTRIBUTION (details in Section 2.2), which distributes the detectors uniformly in the non-self space based on simulated annealing optimization. These two functions will be discussed in the following sections.

2.1 Determining the number of detectors

Let V_d be the volume covered by an individual detector and let $V_{\text{non-self}}$ be the volume of the non-self space. A rough approximation of the number of detectors can be given by:

$$num_{ab} = \frac{V_{\text{non-self}}}{V_d}. \qquad (1)$$

Note that this is a very optimistic approximation since it does not take into account the fact that, in general, it is impossible to cover a given volume with spherical detectors without allowing some overlapping. If overlapping is allowed, the effective covering volume is not anymore the volume of the hypersphere that defines a detector, but a smaller value. We define the covering volume of a detector as the volume of the inscribed hypercube. The main reason to choose this definition is that there is a straightforward way to cover an *n*-dimensional region using hypercubes without holes.

According to the previous discussion, the effective volume covered by a detector d with radius r is defined as:

$$V_d = \left(\frac{2r}{\sqrt{n}}\right)^n. \qquad (2)$$

Using Equations (1) and (2), it is possible to calculate a good approximation of the number of detectors with a given radius needed to cover the non-self space. This will require, however, the knowledge of the volume of the non-self space, which will be addressed in the remaining part of this section.

Calculating the volume of the self (non-self) set The self/non-self space, U, corresponds to the unitary hypercube, $[0, 1]^n$. Clearly, the volume of the self/non-self space is equal to 1.0; therefore, the volume of the non-self space is defined as:

$$V_{\text{non-self}} = 1 - V_{\text{self}}.$$

In most cases, the input to the NS algorithm is a subset of the self set. Thus, in general, the entire volume of the self space is not known. We assume a model of the self set, $\widehat{S}$, that is defined in terms of a set of self samples, S'. The basic assumption in this definition is that an element that is *close enough* to a self sample is considered as self. The closeness is specified formally by a variability threshold, r_{self}, that defines the minimum distance between a self sample and an element x, such that x can be considered part of the self set. The model of the self set, $\widehat{S}$, is defined as follows:

$$\widehat{S} := \left\{x \in U \mid \exists s \in S', \|s - x\| \leq r_{\text{self}}\right\} .$$

We define V_{self} as the volume of $\widehat{S}$, which is calculated as:

$$V_{\widehat{S}} := \int_U \chi_{\widehat{S}}(x)dx \, ,$$

where $\chi_{\widehat{S}}$ corresponds to the characteristic function of the set $\widehat{S}$ defined by

$$\chi_{\widehat{S}}(x) := \begin{cases} 1 \text{ if } x \in \widehat{S} \\ 0 \text{ if } x \notin \widehat{S} \end{cases} .$$

It is possible to produce an estimate of $V_{\widehat{S}}$ using random sampling. The basic idea is to generate a sequence $\{x_i\}_{i=1..m}$ of random samples uniformly distributed in U. The expected value of $\chi_{\widehat{S}}(x_i)$ is

$$\text{E}\left[\chi_{\widehat{S}}(x_i)\right] = \int_U \chi_{\widehat{S}}(x)dx = V_{\widehat{S}};$$

therefore, an estimate of $\text{E}\left[\chi_{\widehat{S}}(x_i)\right]$ is also an estimate of $V_{\widehat{S}}$. As it is well known, a good estimate of the mean of a random variable (expected value) is the mean of a set of samples; so, we use the average of $\left\{\chi_{\widehat{S}}(x_i)\right\}_{i=1..m}$ as an estimate, $\widehat{V_{\widehat{S}}}$, of the self volume:

$$V_{\widehat{S}} \approx \widehat{V_{\widehat{S}}} = \frac{\sum_{i=1}^{m} \chi_{\widehat{S}}(x_i)}{m}. \tag{3}$$

The estimation of a defined integral by averaging a set of random samples is known as *Monte Carlo integration* [2,14]. The main advantage of this method, in contrary to other non-probabilistic methods, is that it is possible to calculate an interval of confidence for the estimated integral. Using the *central limit theorem* [1], it is possible to calculate such interval of confidence as:

$$Pr\left(|\widehat{V_{\widehat{S}}} - V_{\widehat{S}}| < 3\sqrt{\frac{\widehat{V_{\widehat{S}}} - \widehat{V_{\widehat{S}}}^2}{m}}\right) \approx 0.998. \tag{4}$$

```
CALCULATE-INIT-DETECTOR-SET($S'$, $r_{self}$, $r_{ab}$, $\epsilon_{max}$, $init\_iter$)
      $S'$ : set of self samples            $\epsilon_{max}$ : maximum allowed error
  $r_{self}$ : self variability threshold   $m_{min}$ : initial number of iterations
    $r_{ab}$ : detector radius                    $n$ : dimension of the self/non-self space
 1: $num\_hits \leftarrow 0$
 2: $m \leftarrow 0$
 3: Repeat
 4:   $m \leftarrow m + 1$
 5:   $x \leftarrow$uniformly distributed random sample from $[1, 0]^n$
 6:   $y \leftarrow$NEAREST-NEIGHBOR($S'$, $x$)
 7:   If $\|x - y\| \leq r_{self}$
 8:   Then $num\_hits \leftarrow num\_hits + 1$
 9:   EndIf
10:   $\widehat{V_{\widehat{S}}} \leftarrow \frac{num\_hits}{m}$ ▷ Eq. 3
11:   $\epsilon \leftarrow 3\sqrt{\frac{\widehat{V_{\widehat{S}}} - \widehat{V_{\widehat{S}}}^2}{m}}$ ▷ Eq. 4
12: Until $m \geq m_{min}$ and $\epsilon \leq \epsilon_{max}$
13: $num_{ab} \leftarrow \left\lfloor \frac{1 - \widehat{V_{\widehat{S}}}}{\left(\frac{2r_{ab}}{\sqrt{n}}\right)^n} \right\rfloor$ ▷ Eq. 2
14: $D \leftarrow$ Ø
15: Repeat
16:   $x \leftarrow$uniformly distributed random sample from $[1, 0]^n$
17:   $y \leftarrow$NEAREST-NEIGHBOR($S'$, $x$)
18:   If $\|x - y\| \geq r_{self}$
19:   Then $D \leftarrow D \cup \{x\}$
20:   EndIf
21: Until $|D| = num_{ab}$
22: Return $D$
```

Fig. 2. Algorithm to calculate an initial detector set.

Algorithm to calculate an initial set of detectors Now that we know how to calculate the area of the self (non-self) space, it is straightforward to calculate the number of detectors that are needed to cover the non-self space and to generate an initial set of detectors located in the non-self space. The pseudo-code of this algorithm is given below (Figure 2).

The algorithm receives (as input) samples from self (S'), the variability radius of the self (r_{self}), the radius of each detector (r_{ab}), the maximum allowed error (ϵ_{max}), and a minimum number of iterations that have to be performed (m_{min}). The purpose of the last parameter, m_{min}, is to produce a good initial estimate of the error (ϵ) by enforcing a minimum number of iterations. This prevents a premature stop of the algorithm due to a poor initial estimation of ϵ. Notice that the algorithm can be easily modified to receive as input the number of detectors instead of the detector radius (r_{ab}). In that case, line 13 (Figure 2) must be replaced by

$$r_{ab} \leftarrow \sqrt[n]{\frac{1 - \widehat{V_{\widehat{S}}}}{num_{ab}}} \cdot \frac{\sqrt{n}}{2}. \tag{5}$$

2.2 Improving the detector distribution

We describe a procedure to improve the distribution of detectors produced by the CALCULATE-INIT-DETECTOR-SET algorithm (Figure 2) in order to optimize the coverage of the non-self space.

The problem of finding a set with good distribution of detectors can be stated as an optimization problem as follows:

Maximize:

$$V(D) = Volume\{x \in U \mid \exists d \in D, \|x-d\| \leq r_{ab}\}, \tag{6}$$

restricted to:

$$\{s \in S' \mid \exists d \in D, \|s-d\| \leq r_{ab}\} = \text{Ø (not covering of self)}, \tag{7}$$

where,
D : set of detectors with a fixed cardinality, num_{ab},
r_{ab} : detector radius, and
S' : input self set.

The function defined in Equation (6) represents the amount of the self/non-self space covered by a set of detectors, D, which corresponds to the volume covered by the union of hyper-spheres associated with each detector. The restriction specified in Equation (7) tells that no detector should match any self point.

The evaluation of the function $V(D)$ can be a costly process; in fact, the only practical way to compute it is to use a Monte Carlo integration method similar to the one used in the previous section (2.1). Instead, we will use a simplified version of this optimization problem, which we will show, experimentally, to be an equivalent.

Next we describe an optimization algorithm to solve this problem. The technique uses a very well known Monte Carlo based optimization method, *simulated annealing*, which is adapted to solve this particular problem.

Simulated annealing The simulated annealing technique was initially proposed by Kirkpatrick et al. [13] borrowing inspiration from the physical annealing of solids. The physical process can be described as follows: a solid is heated to a high temperature, then, it is slowly made to cool down until some desired properties of the solid are obtained; these properties are related to a low energy state.

In the algorithm, the energy corresponds to the function to minimize, $C(s)$, whose domain is the space of states of a system. The system is randomly perturbed by moving it from the current state, s_i, to a new state, s_j. If $C(s_j) < C(s_i)$, the transition is accepted; otherwise, its acceptance is defined by a random process. The probability of accepting this transition is a function of the temperature: the higher the temperature, the higher the probability of accepting a worse state. This step is repeated a number of times until the system reaches *thermal equilibrium*. This perturbation process is known as the Metropolis algorithm [14,15], and it belongs to a broader class of algorithms called Monte Carlo methods [14].

In our particular problem, we are searching for a set of detectors that optimizes the coverage of non-self space. In consequence, the configuration of the system is given by the coordinates of the detector set. Notice that the number of detectors is fixed (based on the estimate produced by CALCULATE-INIT-DETECTOR-SET, Figure 2), no detectors are created or eliminated in this algorithm.

The original function to optimize corresponds to the volume covered by the detector set (Equation (6)); however, to calculate it can be very costly. Therefore, we need another function which is easier to calculate, and such that its optimization corresponds to the optimization of the covered volume. Intuitively, to maximize the coverage produced by a set of detectors, it is necessary to reduce their overlapping, i.e., to increase the inter-detector distance. The following equation defines an approximate measure of overlapping between two detectors:

$$\text{Overlapping}(d_i, d_j) = e^{\frac{-\|d_i - d_j\|^2}{r_{ab}^2}}. \tag{8}$$

The maximum value, 1, is reached when the distance between the two detectors is 0. When the distance is equal to $2r_{ab}$, the value of the function is very close to 0. Notice that this function can be interpreted as the matching function of the detector.

Based on Equation (8), the amount of overlapping of a set, $D = \{d_1, \ldots, d_{num_{ab}}\}$, of detectors is defined as

$$\text{Overlapping}(D) = \sum_{i \neq j} e^{\frac{-\|d_i - d_j\|^2}{r_{ab}^2}}, \; i, j = 1, \ldots num_{ab}. \tag{9}$$

Now, the question is if minimizing $Overlapping(D)$ is the same as maximizing $V(D)$ (Equation (6)). In general, it is not true; however, we will show in the next section that in the practice they are equivalent.

The original optimization problem includes a restriction that prevents detectors from covering the self (Equation (7)). Simulated annealing does not provide a direct way to include such restrictions; therefore, it is necessary to include a term in the cost function that penalizes configurations which violate this restriction. Then, the function to optimize is defined as follows:

$$C(D) = \text{Overlapping}(D) + \beta \cdot \text{SelfCovering}(D), \tag{10}$$

where, the second term corresponds to the penalization factor for violating the self-covering restriction, and is defined by

$$\text{SelfCovering}(D) = \sum_{s \in S'} \sum_{d \in D} e^{\frac{-\|d - s\|^2}{\left(\frac{r_{ab} + r_{self}}{2}\right)^2}}. \tag{11}$$

Notice that this function is based on the same principle used to define the *Overlapping* function (Equation (9)). Each individual term on the sum measures the amount of matching between a detector and a self element.

The term β in Equation (10) specifies the relative importance of self-covering with respect to the inter-detector overlapping. It controls the amount of penalization in the cost function caused by violating the self-covering restriction.

An advantage of this cost function is that in each step of the algorithm it is not necessary to calculate all the terms in Equations (9) and (10). It is only required to evaluate the terms that involve the detectors affected by the transition (detector movement).

Optimization algorithm for detector distribution The detector distribution algorithm is shown in Figure 3. The main inputs to the algorithm are the initial detector set (generated by the CALCULATE-INIT-DETECTOR-SET algorithm, Figure 2), D; the set of self samples, S'; and the number of iterations, num_{iter}. The shape of detectors (and self elements) is determined by the detector radius, r_{ab}, and the self variability threshold, r_{self}, respectively. The number of iterations on the inner loop (lines 5 to 20) is controlled by the parameter, η_{min}, which expresses the minimum number of accepted transitions as a percentage of the number of detectors. The temperature decay rate, α, and the neighborhood radius decay rate, α_{pert}, control how the temperature and the neighborhood radius are going to be changed in each iteration of the outer loop. Finally, the parameter β specifies the relative importance of covering self points when calculating the cost function.

3 RRNS experimentation

3.1 Overlapping vs non-self coverage

Section 2.2 formulates the problem of detector distribution as an optimization problem corresponding to maximizing the non-self volume covered by a set of detectors ($V(D)$, Equations (6) and (7)). The OPTIMIZE-DETECTOR-DISTRIBUTION algorithm (Figure 3) solves a modified optimization problem: to minimize the function $C(D)$ defined by Equation (10). This function is composed of two terms: one measures the amount of overlapping between detectors and the other penalizes the covering of self points. The main assumption is that minimizing $C(D)$ is approximately equivalent to maximizing $V(D)$. The intuition behind this assumption is that the lesser the overlapping of a set of detectors, the larger the volume they can cover.

Figure 4 shows the evolution of the area covered by a set of detectors and their overlapping, when the OPTIMIZE-DETECTOR-DISTRIBUTION (Figure 3) is applied to an initial set of random detectors in a unitary square. The overlapping, which is the objective function minimized by the algorithm, goes down with the successive iterations. This means that the detectors are moving apart resulting in an increase in the area covered by them, as shown in Figure 4(a). The area curve is not as smooth as the overlapping curve; this can be explained by the fact that the area is estimated (using Monte Carlo integration, $\epsilon = 0.01$), whereas the amount of overlapping is calculated exactly.

This experiment suggests that, in fact, the algorithm is able to maximize the area covered by minimizing the inter-detector overlapping. However, this experiment in a 2-dimensional space is not significant enough to evaluate the algorithmic performance. In order to build a stronger experimental evidence, we performed the following experiment: a random set of detectors is generated close to the center of the unitary hypercube, then the function OPTIMIZE-DETECTOR-DISTRIBUTION (Figure 3) is applied for a given number of iterations, the volume covered and the inter-detector overlapping (Equation

```
OPTIMIZE-DETECTOR-DISTRIBUTION($D$, $r_{ab}$, $S'$, $r_{self}$, $num_{iter}$, $\eta_{coef}$, , $\alpha$, $\alpha_{pert}$, $\beta$)
$D = \{d_1, \ldots, d_{num_{ab}}\}$ : initial detector set          $\eta_{min}$ : minimum accepted transitions %
                          $S'$ : set of self samples                $\alpha$ : Temperature decay rate
                  $num_{iter}$ : number of iterations         $\alpha_{pert}$ : Neighborhood radius decay rate
                      $r_{ab}$ : detector radius                    $\beta$ : Self covering importance coefficient
                    $r_{self}$ : self variability threshold
1: $r_{pert} \leftarrow 2 \cdot r_{ab}$
2: $T \leftarrow$ CALCULATE-INIT-T($D$, $r_{ab}$, $S'$, $r_{self}$, $r_{pert}$, $\beta$)
3: For $i \leftarrow 1$ to $num_{iter}$
4:   $\eta \leftarrow 0$, $steps \leftarrow 0$
5:    Repeat
6:     $index \leftarrow$ random element $\{1, .., num_{ab}\}$
7:     $d \leftarrow$ random element $\{v \in [0,1]^n \mid \|d_{index} - v\| \leq r_{pert}\}$
8:     $\Delta C \leftarrow$ CALCULATE-COST-DIFFERENCE($D$, $index$, $d$, $r_{ab}$, $S'$, $r_{self}$, $\beta$)
9:      If $\Delta C < 0$
10:     Then ▷ accept transition
11:       $\eta \leftarrow \eta + 1$
12:       $d_{index} \leftarrow d$
13:      Else
14:       If $e^{\frac{-\Delta C}{T}} >$ random $[0, 1)$
15:       Then ▷ accept transition
16:         $\eta \leftarrow \eta + 1$
17:         $d_{index} \leftarrow d$
18:        EndIf
19:     EndIf
20:    Until $\eta \geq \eta_{min} \cdot num_{ab}$ or $steps > 2 \cdot \eta_{min} \cdot num_{ab}$
21:    $T \leftarrow \alpha \cdot T$
22:    $r_{pert} \leftarrow \alpha_{pert} \cdot r_{pert}$
23: EndFor
24: Return $D$
```

Fig. 3. Algorithm to optimize the distribution of detectors in order to improve the coverage of non-self space.

(9)) are measured; this process is repeated 30 times, each time starting with a new random set of detectors (around the center).

Figure 5 shows the overlapping-versus-volume graphics corresponding to the data generated by the experiment for space dimension 5 and 10. It is easy to see that there is a clear inverse relationship between the volume covered by a set of detectors and their inter-detector overlapping. As it is shown in Figure 5, the relationship is not necessarily linear; however, it does not affect the algorithmic performance as the results demonstrate that the volume increases monotonically while the amount of overlapping decreases.

3.2 RRNS vs RNS

An interesting question is: how does the new algorithm (RRNS) compare to the previous algorithm (RNS) in terms of the optimization of the volume covered by the set of

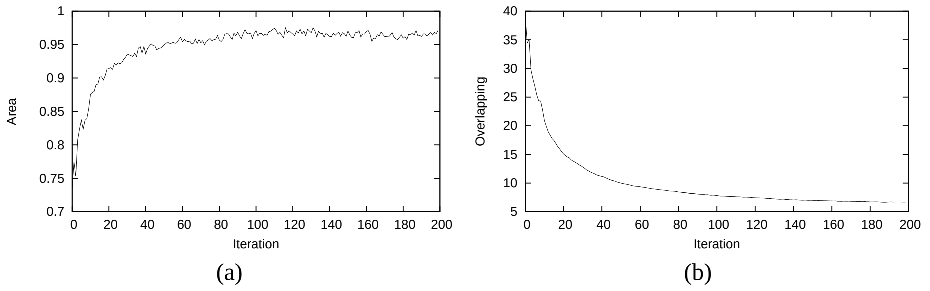

Fig. 4. The RRNS is applied to spread a set of detectors in an unitary square. (a) Progress in the area covered by the detectors. (b) Evolution of the inter-detector overlapping calculated using Equation (9).

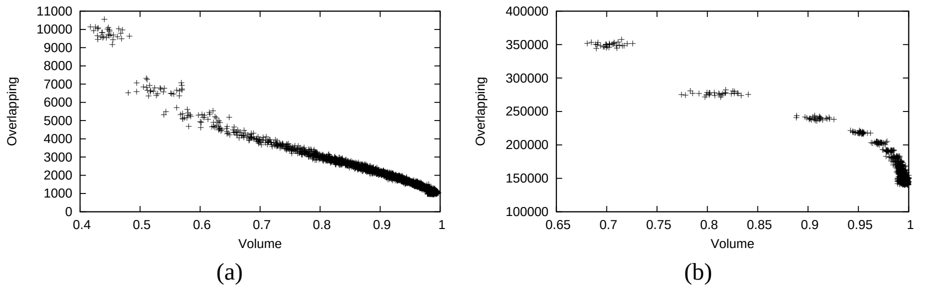

Fig. 5. The graphics show the overlapping-versus-volume relation for a set of detectors produced by the successive application of OPTIMIZE-DETECTOR-DISTRIBUTION function (Figure 3). (a) Dimension = 5. (b) Dimension = 10.

detectors? It is important to take into account that the RNS algorithm was not developed to optimize explicitly the volume or the overlapping. The RNS algorithm is based on heuristic rules that try to move the detectors away from each other and from the self points. An indirect result of this is an increase in the non-self space covered by the set of detectors. Therefore, we expect the RRNS algorithm to perform better than the RNS algorithm in terms of the optimization of the volume covered by the generated set of detectors.

To perform a comparison, we used two data sets based on the Mackey-Glass time series (as described in [9]) having two and four features respectively. Notice that the RRNS is able to calculate the detector radius if the number of detectors is given (Equation (5)); this is not the case for RNS. Therefore, to make a meaningful comparison, we used the detector radius calculated by the RRNS algorithm as input to the RNS algorithm.

Both algorithms are run for a fixed number of iterations. After each iteration, the volume covered by the set of detectors is calculated using a Monte Carlo integration method similar to the one described in the CALCULATE-INIT-DETECTOR-SET algorithm

(Figure 2). In this case, the value of the error is ϵ=0.005. The process is repeated 30 times (i.e. 30 experiments). Figure 6 shows the evolution of the covered volume for each algorithm and for both data sets.

The points in the curve represent the average volume of 30 experiments, and the length of vertical lines correspond to three times the standard deviation. In both cases, the covered volume increases in successive iterations. The RRNS algorithm produces a larger covering volume, as was expected because of its optimization components. The results are encouraging , which suggest that the theoretical foundation of the RRNS also provides a more efficient coverage of the non-self; however, a more extensive testing (with different data sets) is needed in order to assess the real strength of the RRNS algorithm.

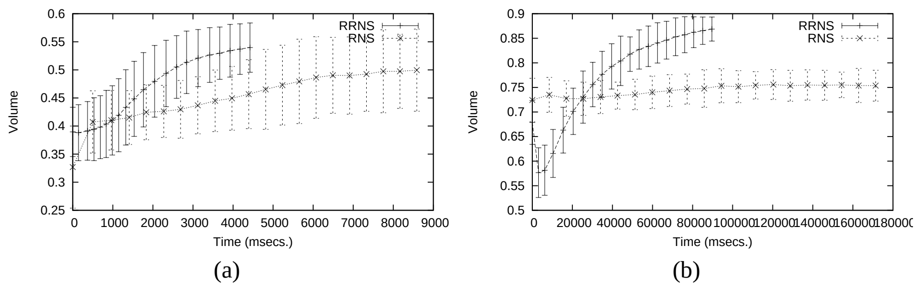

Fig. 6. Evolution of the non-self covered volume against the time for RNS and RRNS algorithms. (a) 2-dimensional data set; (b) 4-dimensional data set.

4 Conclusions

This paper presented a NS algorithm to generate detectors in a real-valued non-self space, called Randomized Real-Valued Negative Selection (RRNS) algorithm. The algorithm is based on Monte Carlo simulation techniques; this gives it the appellative of *randomized*. The algorithm improves the RNS algorithm by providing a mathematical support that facilitates:

- the production of a good estimate of the number of detectors (of a given radius) needed to cover the non-self space, and
- the provision of a guarantee, at least theoretically, that the algorithm can converge to an optimal configuration.

The RRNS algorithm appears to be better than the RNS algorithm in providing a theoretical basis for analyzing its performance. However, this does not mean to claim that it can produce better empirical results. In some cases, heuristic algorithms outperform other algorithms with better theoretical foundation. Preliminary experiments presented in this paper suggest that the RRNS algorithm can offer an improved performance. It is necessary, however, to perform extensive experimentation to measure the real strength of the RRNS algorithm as well as the impact of the improved non-self coverage on the anomaly detection performance.

References

1. H. D. Brunk. *An Introduction to Mathematical Statistics*. Blaisdell Publishing Co., 1965.
2. Computational Science Education project. Introduction to monte carlo methods. Oak Ridge National Laboratory, 1995.
3. L. N. de Castro and J. Timmis. *Artificial Immune Systems: A New Computational Approach*. Springer-Verlag, London, UK, 2002.
4. L. N. De Castro and F. J. Von Zuben. An immunological approach to initialize centers of radial basis function neural networks. In *Proc. of CBRN'01 (Brazilian Conference on Neural Networks)*, pages 79–84, 2001.
5. P. D'haeseleer, S. Forrest, and P. Helman. An immunological approach to change detection: algorithms, analysis and implications. In J. McHugh and G. Dinolt, editors, *Proceedings of the 1996 IEEE Symposium on Computer Security and Privacy*, pages 110–119, USA, 1996. IEEE Press.
6. F. Esponda, S. Forrest, and P. Helman. A formal framework for positive and negative detection schemes. Draft version, July 2002.
7. G. S. Fishman. *Monte Carlo. Concepts, algorithms, and Applications*. Springer-Verlag, 1996.
8. S. Forrest, A. Perelson, L. Allen, and R. Cherukuri. Self-nonself discrimination in a computer. In *Proceedings IEEE Symposium on Research in Security and Privacy*, pages 202–212, Los Alamitos, CA, 1994. IEEE Computer Society Press.
9. F. González and D. Dagupta. Neuro-immune and self-organizing map approaches to anomaly detection: A comparison. In J. Timmis and P. J. Bentley, editors, *Proceedings of the 1st International Conference on Artificial Immune Systems (ICARIS)*, pages 203–211, Canterbury, UK, Sept. 2002. University of Kent at Canterbury Printing Unit.
10. F. Gonzalez, D. Dasgupta, and J. Gomez. The effect of binary matching rules in negative selection. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, July 2003.
11. F. González, D. Dasgupta, and R. Kozma. 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, editors, *Proceedings of the 2002 Congress on Evolutionary Computation CEC2002*, pages 705–710, USA, May 2002. IEEE Press.
12. P. Harmer, G. Williams, P.D.and Gnusch, and G. Lamont. An Artificial Immune System Architecture for Computer Security Applications. *IEEE Transactions on Evolutionary Computation*, 6(3):252–280, June 2002.
13. S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science, Number 4598, 13 May 1983*, 220(4598):671–680, 1983.
14. J. S. Liu. *Monte Carlo Strategies in Scientific Computing*. Springer-Verlag, 2001.
15. N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092, 1953.
16. P. J. M. van Laarhoven and E. H. L. Aarts. *Simulated Annealing: Theory and Applications*. D. Reidel Publishing Company, 1987.