Proceedings of the 2007 IEEE Symposium on
Foundations of Computational Intelligence (FOCI 2007)

A Framework for Evolving Multi-Shaped Detectors in Negative Selection

Sankalp Balachandran, Dipankar Dasgupta, Fernando Nino*, Deon Garrett
Computer Science Department
The University of Memphis
and *Universidad Nacional de Colombia
{sblchndr,dasgupta, jdgarrtt }@memphis.edu, *lfninov@unal.edu.co

Abstract

This paper presents a framework to generate multi-shaped detectors with valued negative selection algorithms (NSA). In particular, detectors can take the form of hyper-rectangles, hyper-spheres and hyper-ellipses in the non-self space. These novel pattern detectors (in the complement space) are evolved using a genetic search (the structured genetic algorithm), which uses hierarchical genomic structures and a gene activation mechanism to encode multiple detector shapes. This genetic search (the Structured GA) allows in maintaining diverse shapes while contributing to the proliferation of best suited detector shapes in expressed phenotype. The results showed that a significant coverage of the non-self space could be achieved with fewer detectors compared to other NSA approaches (using only single-shaped detectors). The uniform representation scheme and the evolutionary mechanism used in this work can serve as a baseline for further extension to use several shapes, providing an efficient coverage of non-self space.

Keywords
Evolutionary Algorithms, Artificial immune systems, Negative selection, Monte Carlo estimation, Computational geometry.

1. Introduction

The negative selection algorithm (NSA) is the first immune-inspired change detection algorithm [2, 12], which originally used a binary representation to encode the self/non-self space. Subsequently, Gonzalez et al. [3] introduced a real-valued representation, called real-valued negative selection (RNS) algorithms to alleviate the scaling issues of binary encoding, while various schemes are being developed to speed up the detector generation process. However, different RNS algorithm is developed to generate one or the other geometric shaped detectors (such as hyper-rectangles [1, 3], hyper-spheres [5] and hyper-ellipses [6]) for covering the non-self space.

The work described in this paper focused on developing a framework for generating multi-shaped detectors with real-valued negative selection algorithm (RNS). These detectors are evolved using an evolutionary algorithm, called Structured GA, which can encode multiple shapes in its genomic structure. In addition, the *matching rule* for each specific detector is expressed by a *membership function* related to that detector shape, which is a function of distance measure (in $\Re^n$).

1.1 Negative Selection Algorithm (NSA)

The Negative Selection Algorithm (NSA), developed by Forrest et al. [12], is based on the principles of self/non-self discrimination in the immune system. It can be summarized as follows:

- Define self as a collection *S* of elements in a feature space *X*, a collection that needs to be monitored. For instance, if *X* corresponds to the space of states of a system represented by a list of features, *S* can represent the subset of states that are considered as normal for the system.
- Generate a set *F* of *detectors*, each of which fails to match any element in *S*.
- Monitor *S* for changes by continually matching the detectors in *F* against *S*. If any detector ever matches, then a change is known to have occurred, as the detectors are designed not to match any representative samples of *S*.

In the original version of a NSA, elements in the shape space and detectors (matching rules) were represented using binary strings [2, 12], but subsequently, some variations of this algorithm using real-valued representation were developed [1, 3, 5]. In RNS representation, elements of the shape-space and detectors are defined in an *n*-dimensional space ($[0, 1]^n$) with Euclidean distance as *matching rules*. The RNS algorithms take as input a set of self samples (represented by *n*-dimensional points) and try to generate a set of detectors covering the non-self space efficiently [1, 4-6].

1.2 Structured GAs

A structured GA (st.GA) is a variation of genetic algorithm developed by Dasgupta [11] in early 1990s. A structured GA incorporates redundant genetic material, which is controlled by a well-defined gene activation mechanism. It utilizes multi-layered genomic structures for the chromosome. All genetic material (currently expressed and currently neutral) is encoded in a hierarchical chromosome. The activation mechanism enables and disables these encoded genes. This encoded redundancy has the potential in maintaining genetic diversity – identified as necessary for dealing with complex problems.

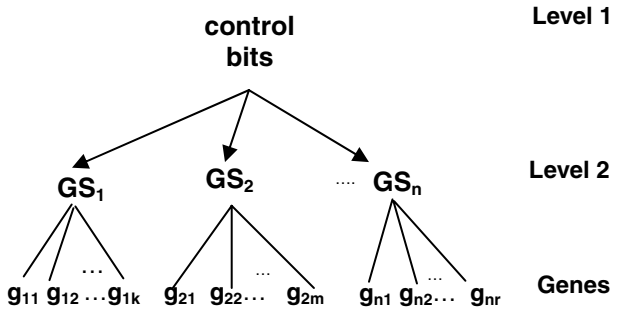

***Figure* 1.** *Representation of a hierarchical chromosome with different gene sets in a St. GA.*

Figure 1 shows a multi-level genomic structure where the higher level control bits activate/deactivate *set of lower level genes* encoding different detector shapes. Therefore, whether a gene will be expressed phenotypically or not is regulated by high level genes.

1.3 Volume estimation using a Monte Carlo Method

In this work, Monte Carlo methods are used for volume estimation, particularly, to estimate the coverage of a set of detectors. Also, this technique is be useful in probabilistically estimating the overlap among detectors with different shapes, which is otherwise cumbersome geometrically. The main aspects of the Monte Carlo method are summarized below (for a more comprehensive explanation see [8, 9]).

Let $X = [0, 1]^n$ be the system state space and $A \subseteq X$ a subset of X, whose volume needs to be computed. Also, a general assumption is that it is hard to compute the volume of A analytically. If x is drawn from a uniform distribution on X, then $P(x \in A) =$ volume of A, denoted as $V(A)$, because $V\left([0,1]^n\right) = 1$. Then the problem of computing $V(A)$ can be seen as the problem of estimating $P(x \in A)$.

Therefore, let U be a random variable uniformly distributed on X, denoted $U \sim$ U(0, 1). Let $U_1, U_2, \ldots U_n$ be a sequence of *independent and identically distributed (iid)* U(0,1) random variables. Then, the sequence $X_1, X_2, ..., X_N$ of random variables, generated as follows, are uniformly *iid* in $[0,1]^n$, denoted $X_i \sim$ U($[0,1]^n$).

$$X_1 = (U_1, \ldots, U_n)$$
$$X_2 = (U_{n+1}, ..., U_{2n})$$
$$\ldots$$
$$X_N = (U_{(N-1)n+1}, ..., U_{nN})$$

In order to estimate the volume of A, generate a sequence $X_1, X_2, ..., X_N$, as defined above. Then, an estimation of the volume of A may be computed as

$\hat{V}(A) = \frac{\left|\{i : X_i \in A\}\right|}{N}$, where $|\cdot|$ denotes the number of points in a set. In other words, the volume of A is estimated as the fraction of points that lie in A. An estimate of the volume of A can also be expressed as $\hat{V}(A) = \frac{\sum_{i=1}^{N} Y_i}{N}$, with $Y_i = I_A(X_i)$, where $I_A(\cdot)$ denotes the indicator function of set A. $Y_1, Y_2, ..., Y_N$ is a sequence of independent Bernoulli random variables, with $P(Y_i = 1)$ equal to $P(X_i) = \int \ldots \int_A dx_1 \ldots dx_n = V(A)$.

The main advantage of this method is that it is possible to calculate a confidence interval for the estimated volume $\hat{V}(A)$ as follows. In order to estimate the volume with a confidence of (1-α), using the Chernoff bound, it can be shown that if

$N \geq \frac{3\ln\left(\frac{2}{\alpha}\right)}{\varepsilon^2 V(A)}$ then

$P\left(\left|V(A) - \hat{V}(A)\right| > \varepsilon V(A)\right) \leq \alpha$. However, as the dimensionality increases $V(A)$ approaches zero exponentially quickly, which will require a sample size exponentially large [13]. Nevertheless, in many practical applications, dimensionality is such that reasonably small sample sizes are sufficient.

2. Framework for evolving negative detectors

The approach developed in this work uses a real-valued representation to characterize the *self/non-self* space and evolves a set of detectors (rules) that can cover the (*non-self*) complementary space. Specifically, the self/non-self

space corresponds to the unit hypercube $[0, 1]^n$, and the detectors are *hyper-shapes* contained in this space, which can be interpreted as anomaly detection rules.

Therefore, a detector is defined in an n-dimensional space as a geometrical shape, such as a hyper-sphere, a hyper-rectangle or a hyper-ellipse. The *matching rule* is expressed by a *membership function* associated to the detector, which is a function of the detector-sample distance (Euclidean or any other [3]). A set of self samples represented by n-dimensional points are given as inputs to the algorithm.

In order to apply an evolutionary algorithm for detector generation, suitable representation for each detector shape is adopted and respective membership functions are defined. In particular, a structured GA with a two level representations (as shown in Figure 1) is used. The level 1 gene either activates or deactivates genes at level 2 which encodes different detector shapes. This provides a generalized way of representing multi-shaped detectors.

The goal of the NS algorithm is to evolve a set of detection rules to cover the non-self space. The iterative process will generate a set of detectors driven by two main goals:

- Minimize overlap with self, and
- Make the detectors as large as possible and keep them separate from each other, in order to maximize the non-self covering.

A niching technique [3, 14] is used along with a st.GA to generate a set of detector shapes as illustrated in Figures 2 and Figure 3. The purpose is to run the complete algorithm multiple times to generate different detectors to cover the entire non-self region. Each run involves the generation of a new detector shape, covering a portion of the non-self region while modifying its raw fitness as per the overlap with the previously selected detectors.

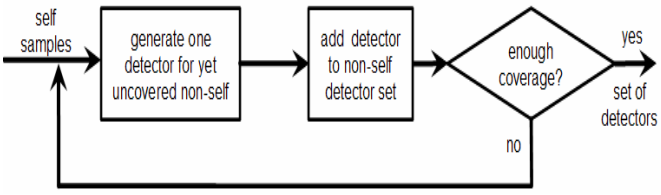

Figure 2. *General framework for detector generation.*

The input to the st.GA is a normalized data set of feature vectors $S=\{x^1, \ldots, x^m\}$, which correspond to normal samples. Each element x^j in S is an n-dimensional vector $x^j=(x^j_1, x^j_2, \ldots, x^j_n)$. Also, a variability parameter (denoted by r) to the entire set of normal samples is considered. Accordingly, r represents the level of variability or allowable deviation in the normal (self) space. This follows the notion that the self sample is never complete, so adding the variation parameter provides an approximation of the self set, given the set of self samples. Figure 3 presents a pseudo-code for the evolutionary detector generation based on a st.GA.

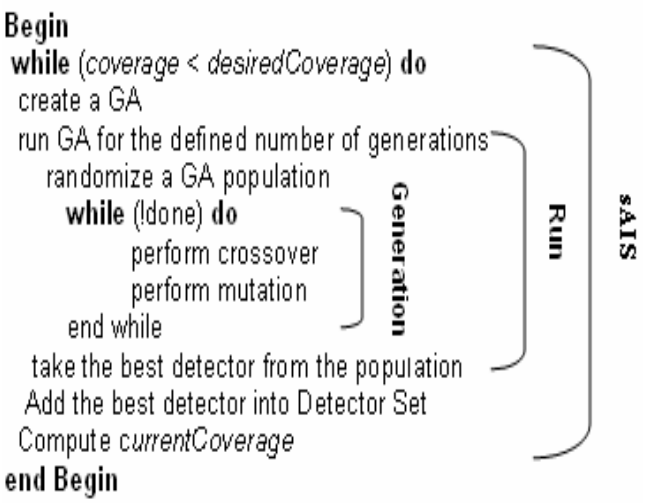

Figure 3. *An Evolutionary approach for detector generation*

2.1 Multi-shaped Detector representation

As shown in Figure 4, each individual (chromosome) expresses a specific shape, namely, hyper-sphere, hyper-rectangle and hyper-ellipse in the phenotypic space. Accordingly, hyper-sphere genes indicate the hyper-sphere n-dimensional center and radius.

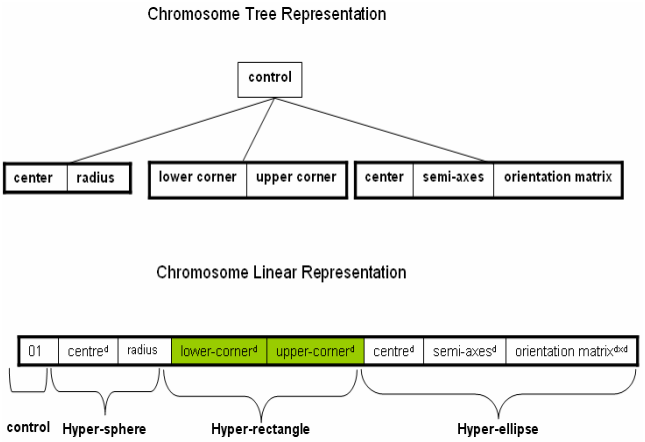

Figure 4. *Example of an encoded chromosome having high level control and low level parameters for three different hyper-shapes; hyper-spheres, hyper-rectangles and hyper-ellipses.*

Similarly, hyper-rectangle genes hold information of the two points that specify the minimum and maximum coordinates in each dimension, analogous to the lower-left

and upper-right corners of a rectangle in two dimensions. On the other hand, hyper-ellipse genes contain its *n*-dimensional center, *n* semi-axes lengths and a square matrix of size *n*×*n* that specifies the orientation of the hyper-ellipse. This detector representation allows having variable-sized detectors which can cover the non-self space.

2.1.1 Hyper-spherical Detectors

A hyper-spherical detector is denoted by $d = (c, R)$, where $c = (c_1, c_2, .., c_n)$ is its center, an *n*-multi-dimensional point, and *R* specifies the radius of the hyper-sphere.

Thus, an *n*-dimensional feature vector $x=(x_1, x_2, \ldots, x_n)$ belongs to the hyper-sphere (matches the detector) if $dist(c,x) < R$, for some particular distance measure $dist(\cdot)$.

In general, the volume of a hyper-sphere [15, 16] is given as [1] $V_n(R) = \frac{(\pi/2)^k (2R)^n}{n!!}$. The calculation of the volume inside the boundaries of the unit hypercube is a difficult problem in the case of edge hyper-spheres [17]. Also, the estimation of overlapped volume of detectors is a challenging problem, specially, when different hyper-shape detectors are used. Therefore, a Monte Carlo technique is used to estimate the volume.

2.1.2 Hyper-rectangular Detectors

A hyper-rectangular detector *d* is defined by two *n*-dimensional points *low* and *high,* i.e., $d = (low, high)$, $low = (low_1, low_2, \ldots, low_n)$ and $high = (high_1, high_2, \ldots, high_n)$, which specify the lower and upper corners of the hyper-rectangle, i.e., $[low_i, high_i]$ specify the range of the *i-t*h coordinate in the hyper-rectangle [1,3]. Thereby, an *n*-dimensional feature vector $x= (x_1, x_2, \ldots, x_n)$ will lie in the hyper-rectangle if $x_i \in [low_i, high_i]$ for each $i=1,2,\ldots, n$. Also, the volume of the subspace represented by a detector *d* is computed as $\prod_{i=1}^{n}(high_i - low_i)$.

2.1.3 Hyper-ellipse Detectors

A hyper-ellipse detector is represented in a general form, considering orientation of the semi-axes in any direction [6]:

$$(x-\omega)^T A(x-\omega) = 1$$

where $A = V\Lambda V^T$ and ω is the center of the hyper-ellipse, $\Lambda=(\lambda_{i,j})$ is an $n\times n$ diagonal matrix such that $\lambda_{i,i} = 1/\ell_i^2$, whose entries are eigenvalues associated with the eigenvectors in *V*; ℓ_i is the length of i-th semi-axis; V is an $n\times n$ matrix whose columns are orthonormal eigenvectors of *A*. Any *n*-dimensional point *x* satisfying the equation above lies on the surface of the hyper-ellipse;

In addition, *x* is inside the hyper-ellipse if $(x-\omega)^T A(x-\omega) < 1$. In 2-dimensions, the general equation above could be easily represented as:

$$\begin{pmatrix} x_1-\omega_1 & x_2-\omega_2 \end{pmatrix} \times \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \times \begin{pmatrix} 1/l_1^2 & 0 \\ 0 & 1/l_2^2 \end{pmatrix} \times \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \times \begin{pmatrix} x_1-\omega_1 \\ x_2-\omega_2 \end{pmatrix} = 1$$

$$(x-\omega)^T \times V \times \Lambda \times V^T \times (x-\omega) = 1$$

where θ is the orientation angle of the hyper-ellipse.

2.2 Fitness Evaluation

The fitness calculation is to evaluate the quality of an individual in a population. Based on their fitness, individuals may be selected as part of the next generation. Here, the fitness evaluation is guided by two main objectives: to cover as large non-self space as possible, while not covering any self point, and the overlap among the detectors should be kept at minimum. Accordingly, detectors with large volume should be favored over smaller ones.

Also, a *membership function* associated to each detector that establishes whether a point lies inside a shape (or not), is used to compute a detector's fitness. This function will depend on the detector's shape, and is defined based on a distance measure (Euclidean, Minkowski, Mahalanobis or any other). Minkowski distance of order n is defined as the distance between two points *x* and *y*, which is defined as

$$dist(y,x) = \left(\Sigma |y_i - x_i|^n \right)^{1/n}.$$

Let *D* be a set of detectors; each *d* in *D* is a subset of *N*. In this work, it is assumed that *D* will provide a better coverage of *N* if the volume of *D* is as large as possible, while minimizing the overlap of detectors on *D* with *S*. Thus, the goal of the evolutionary search may be expressed in a more formal way as

$$\max \boldsymbol{vol}\left(\bigcup_{\boldsymbol{d}\in \boldsymbol{D}} \boldsymbol{d} \right) \quad \text{and}$$

[1] As per the definition of the Gamma Function in terms of factorials (the notation being k! = 1x2x3x … xk), the coefficient of R^n in the above is: $\pi^k/k!$ with k=n/2 when n is even, or $2^n\pi^k k!/n!$ with k=(n-1)/2 when n is odd

$$\min \boldsymbol{vol}\left(\left(\bigcup_{d\in D}\boldsymbol{d}\right)\cap \boldsymbol{S}\right).$$

Accordingly, proper reward and penalty functions are defined within the detector fitness function. A detector is rewarded for the total space covered, while it will be penalized if it overlaps with other detectors that were found in the previous runs of the GA, and it will also be penalized for all the self points that it covers. Then, the fitness function is defined as

$$fitness(D) = effective\text{-}coverage(D) - C(m)\times m,$$

where m is the number of self points that lie in D, $C(m)$ is a factor that is computed taking into account the number of self points that the detector D overlaps.

The *effective coverage* is computed as the proportion of the non-self set that the detector covers and is not yet covered by other detectors. On the other hand, $C(m)$ is defined in such a ways that a detector suffers a stronger penalization as its overlap with self sample set increases. Thus, $C(m)$ is defined as

$$C(m) = \tanh\left(\frac{m}{\ln|S|}\right),$$ if $m>0$, and $C(m)=0$ if m is equal to 0. Therefore, $C(m)$ gets higher as the number of self points increases.

Estimating Detector Overlap: In order to evaluate the overlap between detectors, a Monte Carlo technique is used. Given two detectors D and E, a sequence of N random points uniformly distributed on $[0,1]^n$ is generated. Then, a test is performed on these points to check whether they belong to D and E simultaneously. The proportion of points that pass this test will give an estimate of the overlap between detectors D and E. Therefore, the *effective-coverage* of a detector D is estimated as

$effective\text{-}coverage(D) = \hat{V}(D) - \beta$, where β is an estimate of the net overlap of D with the other detectors, computed via the Monte Carlo Method, where $\hat{V}(D)$ gives an estimate of the volume of D.

Checking Self Overlap: In order to detect the overlap of a detector with a self-sample, its overlap with a hyper-sphere of radius r around the self point is considered. In the case of a hyper-rectangle shaped detector, this is determined by checking if the circumscribed hypercube around the self hyper-sphere intersects the detector. For a hyper-sphere shaped detector with center c and radius R, there is an overlap with a self point x, if the distance between c and x is less than $(R+r)$.

Figure 5 describes how to check whether a hyper-ellipse detector overlaps a self sample, for a defined self variation threshold r. In the figure, P_2 is the self sample. If P_2 does not lie in the hyper-ellipse then the algorithm checks whether P_3 lies inside the hyper-ellipse, as depicted below. P_1 is the hyper-ellipse's center and $P_3 = P_2 - r\left(\frac{P_2 - P_1}{\|P_2 - P_1\|}\right)$, where $\|\cdot\|$ denotes the norm of a vector.

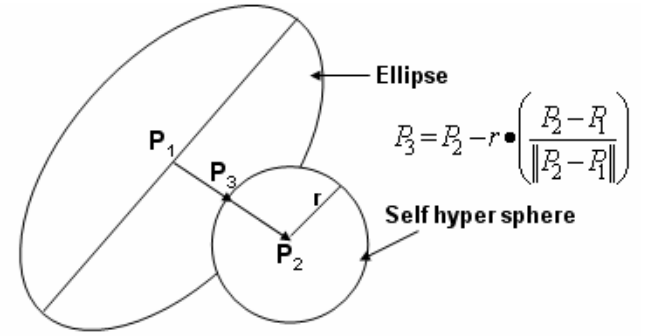

Figure 5*. Determining the overlap of a hyper-ellipse detector with the self set.*

Shapiro's work [6] mentions several measures of convergence checks. One suggested convergence check is to determine if the user defined coverage value has been reached. Another way to check convergence is by checking the change in the standard deviation of the population fitness, which is decreased by certain percentage at a predefined rate [3,4,6].

At the end of this post-processing of the detector list, the net coverage (*Cov*) of the non-self space has been estimated. If *Cov* has reached a desired value specified by the terminating condition, the algorithm stops and the current detector set is returned.

Thus, an evolutionary search is used to evolve 'good' detectors that cover the non-self space (Figures 8-11). The fitness of a detector rule is determined by the number of self samples it covers, its hyper-volume and the overlap with other rules (detectors). Therefore, the goal is to find a set of solutions that collectively cover the entire non-self space, rather than a single solution. Accordingly, a penalization factor is computed for each detector based on the number of self samples that it overlaps.

3. Experimentation

Experiments are conducted to compare the performance of multi-shaped detectors with single shaped detectors generated with RNSs using some synthetic data. These experiments intend to answer the following questions: does

the designed evolutionary framework successfully generate a hyper-shaped detector set? Is this new approach comparable to other existing RNS approaches in terms of detection rate and coverage? What is the role of parameters in controlling the evolution process? To address these questions, some synthetic data sets are used; these datasets form a star-shaped and multi-cluster in a unit square as shown in Figures 6 and Figure 7. The entire self set in each case consists of 1000 points, and assumed as the normal (or self) data sets.

The training data consists of 50% of the entire self set. The test data consists of 1000 points of which 50% is taken from the self data not in training data and the remaining 50% is randomly generated in the complement space.

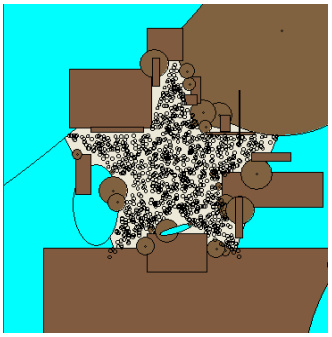

Figure *6. Multi shaped detectors, ~ 97.5% coverage, 0.01 Self Threshold*

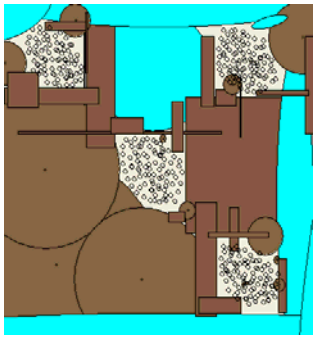

Figure *7. Multi shaped detectors, ~ 94% coverage, 0.01 Self Threshold*

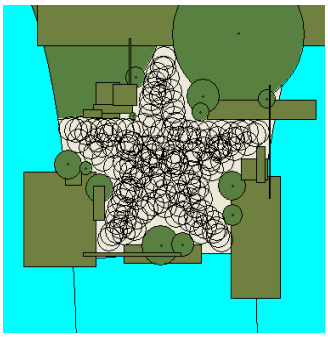

Figure *8. Multi shaped detectors, ~ 98.75% coverage, 0.05 Self Threshold*

Figure *9. Multi shaped detectors, ~ 97.9% coverage, 0.05 Self Threshold*

3.1 Experimental Setting

As mentioned before, the genetic algorithm evolves a single detector in one complete run, and a sequential niching scheme is involved. This work made use of the reproduction of candidate detectors with *n*-point crossover and tournament selection. Also, a Monte Carlo technique is employed to perform overlap and coverage estimation after each run and penalize/discard incompetent detectors.

Statistics of 30 independent trials were collected during the detector generation process for each data set. The average and standard deviation of the coverage in this case are recorded.

The following parameters were used in the experiments:

maximum iterations per run=100; *convergence check between runs*=5; *convergence threshold*=0.05, *self threshold*=0.01-0.07; *Population size*=40; *selection strategy*=tournament with size 2; *Crossover rate*: 0.4 – 0.6; *general mutation rate*: 0.03 – 0.05; *orientation mutation rate*: {0.1, 0.2}; *matrix swap mutation rate*=0.5, and *control code mutation rate*= 0.2.

In the Monte Carlo implementation, in order to achieve a 95% (0.95) confidence level (i.e. α equal to 0.05) and a standard error ε equal to 0.01, at least 50,000 sample points need to be generated. Also to check the convergence of the algorithm, an evaluation is done using the Monte Carlo estimation technique every 10 runs (i.e., λ_c =10)

In addition, the following parameters were used in the evolutionary search.

- sequential niching: $\max_{attempts}$ = 5
- elitism rate = 0.05, cull rate=0.01, Δ = 0.1.

3.2 Results and Analysis

Figures 6 and 7 show the area actually covered by the evolved detector set (of different shapes and sizes) that reached 97% coverage and 94% respectively for a self threshold of 0.01. We observed that varying self threshold values provide different coverage detector coverage, for example, non-self coverage is increased to ~98% when a self threshold of 0.05 is used (Figures 8 and 9). It is also observed that a smaller self radius would result in a high detection rate but high false alarm as well (as observed in [5]), while a larger self radius would result in low detection rate and low false alarm rate. Thus the self radius has a pronounced effect on detection rates and alarm rates.

Figure 10 and Figure 11 show comparative results of running different single-shaped RNS methods and the proposed mixed-shaped approach on synthetic datasets. These figures show ROC curves [10] indicating the performance of the above-mentioned approaches for two synthetic datasets. In particular, graphs show the trade-off between detection rate and false alarm rate (commonly used to measure the accuracy of classifiers). In both cases, mixed shape detectors produced high detection rates even for small false alarm rates as compared to other single shaped detectors.

The variation of the number of detectors with the change in self threshold indicates that the mixed shaped

detectors have an overall good performance in terms of the detector number, while the hyper-ellipse detectors are slightly better in some cases.

However, Figures 12 and 13 show that the coverage attained by mixed shape detectors are comparatively better and faster as compared to single shaped detectors, except hyper-ellipses. Thus, the mixed shape detectors on an average cover the non self space better and thereby have a higher detector to coverage ratio as compared to other single shaped detector algorithms. The rate of change of coverage with respect to the sequential GA run for different algorithms is shown in Figures 12 and 13. In all cases, the mixed shape detectors have a better coverage rate and also the maximum coverage at the end of their respective evolutionary runs. However, in some cases the hyper-ellipse detectors also have better increase in coverage rates but still unable to attain the total maximum coverage as compared to the mixed one. These figures also show that the deviations in results are slightly higher for mixed shaped detectors and hyper-sphere shaped detectors although these are within maximum limits of ± 0.11 of the range.

4. Conclusions

A general framework to evolve multi-shaped anomaly detectors is proposed, and studied. It combines the potential of several detector shapes, namely hyper-ellipses, hyper-spheres and hyper-rectangles and a structured genetic algorithm to generate multiple detector shapes. Monte Carlo (MC) estimation technique is integrated in the negative selection algorithm (NSA) to ensure enough coverage of non-self space by an evolving detector set. This makes the negative selection more reliable while eliminating the need for detector adjustment. The MC detector volume estimation technique may also be applied to any type of detectors and detection mechanism as long as a membership function of a sample point for each detector shape is defined.

Hyper-shapes generated using an evolutionary approach are a considerable improvement over other geometrical shapes previously investigated [3, 5, 6]. Their accuracy is comparable to hyper-ellipses [6], and better than hyper-rectangles [1] and hyper-spheres [5]. In the paper by Ji and Dasgupta [5], the authors concluded that detector set generated by V-detector is more reliable because the expected coverage (instead of arbitrary detector number) can be achieved by the RNS algorithm. The current work took advantage of using variable-sized detectors with variable geometrical shapes to better cover holes, while small number of V-detectors reduces space requirement and access time.

The use of variable shape and size detectors can help to address the high dimension problems (Curse of Dimensionality) in real-valued space (as reported in [17]) along with the choice of different distance measures in this work. Moreover, we only used two synthetic datasets for our experiments, but further study should include different distribution of self data.

It is to be noted that the current work focused on developing a unified framework for generating multi-shaped detectors in real-valued negative selection algorithm (RNS). Preliminary studies showed that the multi-shaped detectors can provide better coverage compared to any single-shaped detectors. However, other issues related to negative selection algorithms, and its comparison with existing anomaly detection methods is beyond the scope of this work but will be investigated in the future.

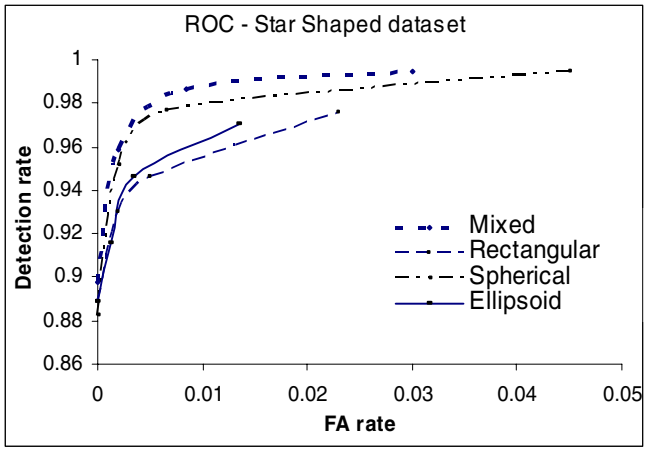

***Figure* 10.** *Comparing all shape strategies for Star dataset*

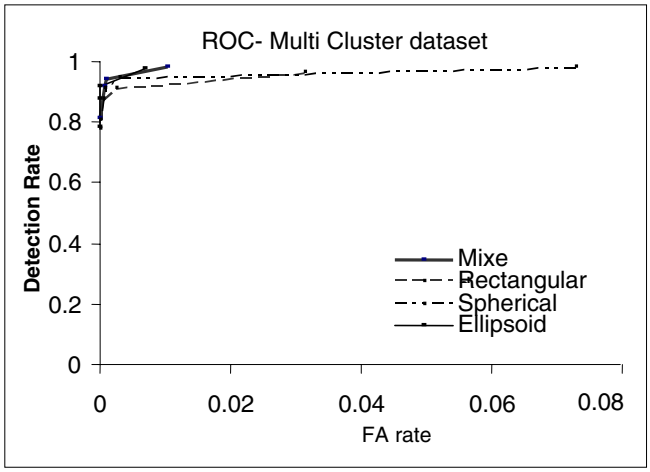

***Figure* 11.** *Comparing all shape strategies: Multi-Cluster*

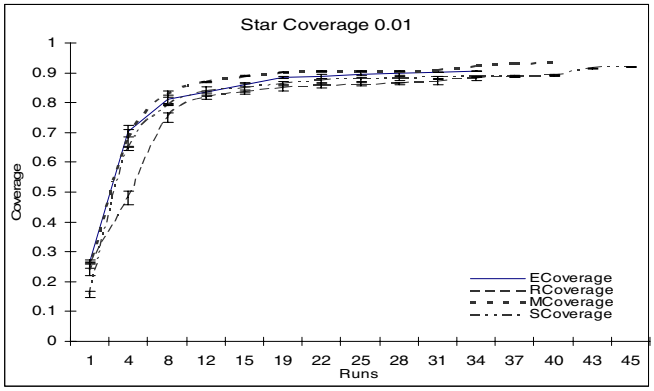

***Figure* 12.** *Rate of change of Coverage with the Runs for self-threshold 0.01 in Star shaped self*

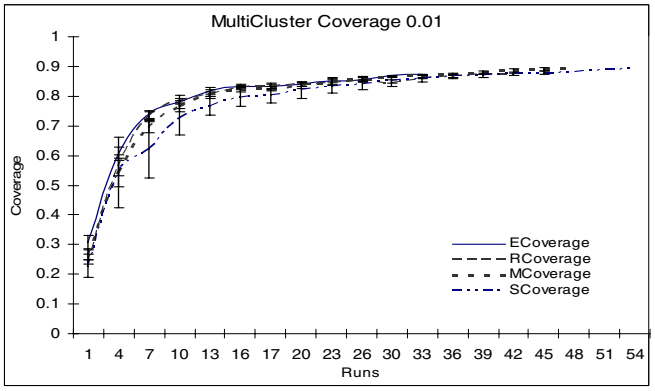

***Figure* 13.** *Rate of change of Coverage with the Runs for self-threshold 0.01 in Multi-Cluster shaped self.*

References

[1] D. Dasgupta and F. González. *An Immunity-Based Technique to Characterize Intrusions in Computer Networks.* IEEE Transactions on Evolutionary Computation, 6(3):281–291, June 2002.

[2] P. D'haeseleer, S Forrest, P. Helman, *An immunological approach to change detection: algorithms, analysis and implications***.** In the Proceedings of IEEE Symposium on Security and Privacy, pp. 110 - 119. June, 1996.

[3] F. González. *A study of Artificial Immune Systems Applied to Anomaly Detection*. PhD. Dissertation, May 2003.

[4] L. J. Gonzalez, James Cannady. *A Self Adaptive Negative Selection Approach for Anomaly Detection*. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO) Seattle, Washington June 26-30, 2004.

[5] Z. Ji and Dipankar Dasgupta. *Real-Valued Negative Selection using Variable-Sized Detectors*. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO) Seattle, Washington June 26-30, 2004.

[6] J. M. Shapiro, G. B. Lamont, Gilbert L. Peterson. *An Evolutionary Algorithm to Generate Hyper-Ellipse Detectors for Negative Selection.* In proceedings of Genetic and Evolutionary Computation Conference (GECCO), Washington, D.C., June 25-29, 2005.

[7] C. L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998. Available at http://www.ics.uci.edu/~mlearn/MLRepository.html.

[8] "Computational Science Education project. Introduction to Monte Carlo methods," Oak Ridge National Laboratory, 1995. Available at: http://csep1.phy.ornl.gov/mc/mc.html

[9] J. S. Liu, *Monte Carlo Strategies in Scientific Computing*. Springer-Verlag, 2001.

[10] F. Provost, T. Fawcett, and R. Kohavi. *The case against accuracy estimation for comparing induction algorithms*. In the Proceedings of 15th International Conference On Machine Learning, pp. 445-453, San Francisco, CA, 1998. Morgan Kaufmann.

[11] D. Dasgupta and D. R McGregor. *sGA: A structured Genetic Algorithm.* Research Report IKBS-11-93, April 1993.

[12] S. Forrest, A.S. Perelson, L. Allen, R. Cherukuri. *Self-nonself discrimination in a computer*. In the proceedings of the 1994 IEEE Symposium on Research in Security and Privacy, IEEE Computer Society Press. 1994.

[13] M. .Mitzenmacher, E. Upfal, *Probability and Computing-Randomized Algorithms and Probabilistic Analysis***.** Cambridge University Press, pp. 252-254, 2005.

[14] W. S. Mahfoud. *"A comparison of parallel and sequential niching methods"*. In L. J. Eshelman, editor, 6th Int. Conf. on Genetic Algorithms, pages 136--143. Morgan--Kaufmann, 1995.

[15] Team, PAL Core Development. "PAL Project: Gamma Function", October 2004.

[16] http://home.att.net/%7Enumericana/answer/functions.htm#gamma

[17] Thomas Stibor, Jonathan Timmis, Claudia Eckert. *A Comparative Study of Real-Valued Negative Selection to Statistical Anomaly Detection Techniques*. In the proceedings of the 4th International Conference on Artificial Immune Systems (ICARIS) pp. 262-275, Sept. 2005.