The Effect of Binary Matching Rules in Negative Selection

Fabio González[1], Dipankar Dasgupta[2], and Jonatan Gómez[1]

[1] Division of Computer Science, The University of Memphis , Memphis TN 38152
and Universidad Nacional de Colombia, Bogotá, Colombia
{fgonzalz,jgomez}@memphis.edu
[2] Division of Computer Science, The University of Memphis, Memphis TN 38152
dasgupta@memphis.edu

Abstract. Negative selection 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 (from given normal samples). The negative selection algorithm generally uses binary matching rules to generate detectors. The purpose of the paper is to show that the low-level representation of binary matching rules is unable to capture the structure of some problem spaces. The paper compares some of the binary matching rules reported in the literature and study how they behave in a simple two-dimensional real-valued space. In particular, we study the detection accuracy and the areas covered by sets of detectors generated using the negative selection algorithm.

1 Introduction

Artificial immune systems (**AIS**) is a relatively new field that tries to exploit the mechanisms present in the biological immune system (**BIS**) in order to solve computational problems. There exist many AIS works [5,8], but they can roughly be classified into two major categories: techniques inspired by the self/non-self recognition mechanism [12] and those inspired by the immune network theory [9,22].

The *negative selection* (**NS**) algorithm was proposed by Forrest and her group [12]. This algorithm is inspired by the mechanism of T-cell maturation and self tolerance in the immune system. Different variations of the algorithm have been used to solve problems of anomaly detection [4,16], fault detection [6], to detect novelties in time series [7], and even for function optimization [3].

A process that is of primary importance for the BIS is the antibody-antigen matching process, since it is the basis for the recognition and selective elimination mechanism that allows to identify foreign elements. Most of the AIS models implement this recognition process, but in different ways. Basically, antigens and antibodies are represented as strings of data that correspond to the sequence of aminoacids that constituting proteins in the BIS. The matching of two strings is determined by a function that produces a binary output (match or not-match).

The binary representation is general enough to subsume other representations; after all, any data element, whatever its type is, is represented as a sequence of bits in the memory of a computer (though, how they are treated may differ). In theory, any matching

E. Cantú-Paz et al. (Eds.): GECCO 2003, LNCS 2723, pp. 195–206, 2003.

rule defined on a high-level representation can be expressed as a binary matching rule. However, in this work, we restrict the use of the term *binary matching rule* to designate those rules that take into account the matching of individual bits representing the antibody and the antigen.

Most works on the NS algorithm have been restricted to binary matching rules like *r*-contiguous [1,10,12]. The reason is that efficient algorithms that generate detectors (antibodies or T-cell receptors) have been developed, exploiting the simplicity of the binary representation and its matching rules [10]. On the other hand, AIS approaches inspired by the immune network theory often use real vector representation for antibodies and antigens [9,22], as this representation is more suitable for applications in learning and data analysis. The matching rules used with this real-valued representation are usually based on Euclidean distance, (i.e. the smaller the antibody-antigen distance, the more affinity they have).

The NS algorithm has been applied successfully to solve different problems; however, some unsatisfactory results have also been reported [20]. As it was suggested by Balthrop et al. [2], the source of the problem is not necessarily the NS algorithm itself, but the kind of matching rule used. The same work [2] proposed a new binary matching rule, *r*-chunk matching (Equation 2 in Section 2.1), which appears to perform better than *r*-contiguous matching.

The starting point of this paper is to address the question: do the low-level representation and its matching rules affect the performance of NS in covering the non-self space? This paper provides some answers to this issue. Specifically, it shows that the low-level representation of the binary matching scheme is unable to capture the structure of even simple problem spaces. In order to justify our argument, we use some of the binary matching rules reported in the literature and study how they behave in a simple bi-dimensional real space. In particular, we study the shape of the areas covered by individual detectors and by a set of detectors generated by the NS algorithm.

2 The Negative Selection Algorithm

Forrest et al. [12] developed the NS algorithm based on the principles of self/non-self discrimination in the BIS. The algorithm can be summarized as follows (taken from [5]):

- Define self as a collection S of elements in a representation space U (also called self/non-self space), a collection that needs to be monitored.
- Generate a set R of *detectors*, each of which fails to match any string in S.
- Monitor S for changes by continually matching the detectors in R against S.

2.1 Binary Matching Rules in Negative Selection Algorithm

The previous description is very general and does not say anything about what kind of representation space is used or what the exact meaning of *matching* is. It is clear that the algorithmic problem of generating good detectors varies with the type of representation space (continuous, discrete, hybrid, etc.), the detector representation, and the process that determines the matching ability of a detector.

A binary matching rule is defined in terms of individual bit matchings of detectors and antigens represented as binary strings. In this section, some of the most widely used binary matching rules are presented.

***r*-contiguous matching.** The first version of the NS algorithm [12] used binary strings of fixed length, and the matching between detectors and new patterns is determined by a rule called *r*-contiguous matching. The binary matching process is defined as follows: given $x = x_1x_2...x_n$ and a detector $d = d_1d_2...d_n$,

$$d \text{ matches } x \equiv \exists i \leq n - r + 1 \text{ such that } x_j = d_j \text{ for } j = i, ..., i + r - 1, \quad (1)$$

that is, the two strings match if there is a sequence of size *r* where all the bits are identical. The algorithm works in a generate-and-test fashion, i.e. random detectors are generated; then, they are tested for self-matching. If a detector fails to match a self string, it is retained for novel pattern detection.

Subsequently, two new algorithms based on dynamic programming were proposed [10], the *linear* and the *greedy* NS algorithm. Similar to the previous algorithm, they are also specific to binary string representation and *r*-contiguous matching. Both algorithms run in linear time and space with respect to the size of the self set, though the time and space are exponential on the size of the matching threshold, *r*.

***r*-chunk matching.** Another binary matching scheme called *r*-chunk matching was proposed by Balthrop et al. [1]. This matching rule subsumes *r*-contiguous matching, that is, any *r*-contiguous detector can be represented as a set of *r*-chunk detectors. The *r*-chunk matching rule is defined as follows: given a string $x = x_1x_2...x_n$ and a detector $d = (i, d_1d_2...d_m)$, with $m \leq n$ and $i \leq n - m + 1$,

$$d \text{ matches } x \equiv x_j = d_j \text{ for } j = i, ..., i + m - 1, \quad (2)$$

where i represents the position where the *r*-chunk starts. Preliminary experiments [1] suggest that the *r*-chunk matching rule can improve the accuracy and performance of the NS algorithm.

Hamming distance matching rules. One of the first works that modeled BIS concepts in developing pattern recognition was proposed by Farmer et al. [11]. Their work proposed a computational model of the BIS based on the idiotypic network theory of Jerne [19], and compared it with the *learning classifier system* [18]. This is a binary model representing antibodies and antigens and defining a matching rule based on the Hamming distance. A Hamming distance based matching rule can be defined as follows: given a binary string $x = x_1x_2...x_n$ and a detector $d = d_1d_2...d_n$,

$$d \text{ matches } x \equiv \sum_i \overline{x_i \oplus d_i} \geq r, \quad (3)$$

where $\oplus$ is the exclusive-or operator, and $0 \leq r \leq n$ is a threshold value.

Different variations of the Hamming matching rule were studied, along with other rules like *r*-contiguous matching, statistical matching and landscape-affinity matching [15]. The different matching rules were compared by calculating the signal-to-noise ratio and the function-value distribution of each matching function when applied to a randomly generated data set. The conclusion of the study was that the Rogers and Tanimoto (**R&T**) matching rule, a variation of the Hamming distance, produced the best performance. The R&T matching rule is defined as follows: given a binary string $x = x_1x_2...x_n$ and a detector $d = d_1d_2...d_n$,

$$d \text{ matches } x \equiv \frac{\sum_i \overline{x_i \oplus d_i}}{\sum_i \overline{x_i \oplus d_i} + 2\sum_i x_i \oplus d_i} \geq r, \tag{4}$$

where $\oplus$ is the exclusive-or operator, and $0 \leq r \leq 1$ is a threshold value.

It is important to mention that a good detector generation scheme for this kind of rules is not available yet, other than the exhaustive generate-and-test strategy [12].

3 Analyzing the Shape of Binary Matching Rules

Usually, the self/non-self space (U) used by the NS algorithm corresponds to an abstraction of a specific problem space. Each element in the problem space (e.g. a feature vector) is mapped to a corresponding element in U (e.g. a bit string).

A matching rule defines a relation between the set of detectors[1] and U. If this relationship is mapped back to the problem space, it can be interpreted as a relation of affinity between elements in this space. In general, it is expected that elements that are matched by the same detector have some common property. So, a way to analyze the ability of a matching rule to capture this 'affinity' relationship in the problem space is to take the subset of U corresponding to the elements matched by a specific detector, and map this subset back to the problem space. Accordingly, this set of elements in the problem space is expected to share some common properties.

In this section, we apply the approach described above to study the binary matching rules presented in section 2.1. The problem space used corresponds to the set $[0.0, 1.0]^2$. One reason for choosing this problem space is that multiple problems in learning, pattern recognition, and anomaly detection can be easily expressed in an *n*-dimensional real-valued space. Also, it makes easier to visualize the *shape* of different matching rules.

All the examples and experiments in this paper use a self/non-self space composed of binary strings of length 16. An element (x, y) in the problem space is mapped to the string $b_0, ..., b_7, b_8, ..., b_{15}$, where the first 8 bits encode the integer value $\lfloor 255 \cdot x + 0.5 \rfloor$ and the last 8 bits encode the integer value $\lfloor 255 \cdot y + 0.5 \rfloor$. Two encoding schemes are studied: conventional binary representation and Gray encoding. Gray encoding is expected to favor binary matching rules, since the codifications of two consecutive numbers only differs by one bit.

[1] In some matching rules, the set of detectors is same as U (e.g. *r*-contiguous matching). In other cases, it is a different set that usually contains or extends U (e.g. *r*-chunk matching).

Figure 1 shows some typical shapes generated by different binary matching rules. Each figure represents the area (in the problem space) covered by one detector located at the center, (0.5,0.5) (1000000010000000 in binary notation). In the case of r-chunk matching, the detector does not correspond to an entire string representing a point on the problem space, rather, it represents a substring (chunk). Thus, we chose an *r*-chunk detector that matches the binary string corresponding to (0.5,0.5), ****00001000****. The area covered by a detector is drawn using the following process: the detector is matched against all the binary strings in the self/non-self space; then, all the strings that match are mapped back to the problem space; finally, the corresponding points are painted in gray color.

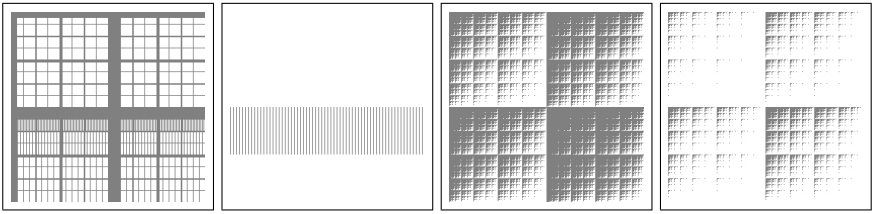

Fig. 1. Areas covered in the problem space by an individual detector using different matching rules. The detector corresponds to 1000000010000000, which is the binary representation of the point (0.5,0.5). (a) *r*-contiguous matching, $r = 4$, (b) *r*-chunk matching, *d* = ****00001000****, (c) Hamming matching, $r = 8$, (d) R&T matching, $r = 0.5$.

The shapes generated by the *r*-contiguous rule (Figure 1(a)) are composed by vertical and horizontal stripes that constitute a grid-like shape. The horizontal and vertical stripes correspond to sets of points having identical bits at least at *r* contiguous positions in the encoded space. Some of these points, however, are not close to the detector in the decoded (problem) space. The *r*-chunk rule generates similar, but simpler shapes (Figure 1(b)). In this case, the area covered is composed of vertical or horizontal sets of parallel strips. The orientation depends on the position of the r-chunk: if it is totally contained in the first eight bits, the strips are vertically going from top to bottom; if it is contained on the last eight bits, the strips are oriented horizontally; finally, if it covers both parts, it has the shape shown in Figure 1(b).

The area covered by Hamming and R&T matching rules has a fractal-like shape, shown in Figure 1(c) and 1(d), i.e. it exhibits self-similarity. It is composed of points that have few interconnections. There is no significant difference between the shapes generated by the R&T rule and those generated by the Hamming rule, which is not a surprise, considering the fact that the R&T rule is based on Hamming distance.

The shape of the areas covered by *r*-contiguous and *r*-chunk matching is not affected by the change in codification from binary to Gray (as shown in Figures 2(a) and 2(b)). This is not the case with the Hamming and the R&T matching rule (Figures 2(c) and

2(d)). The reason is that the Gray encoding represents consecutive values using bit strings with small Hamming distance.

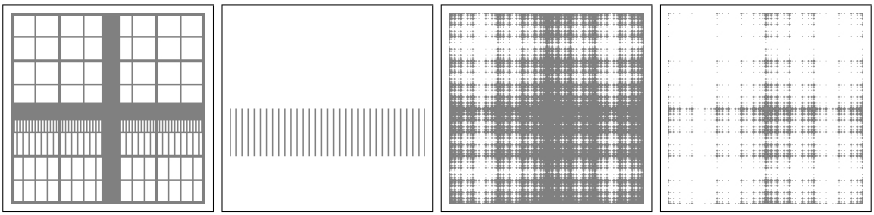

Fig. 2. Areas covered in the problem space by an individual detector using Gray encoding for the self/non-self space. The detector corresponds to 1100000011000000, which is the Gray representation of the point (0.5,0.5). (a) r-contiguous matching, $r = 4$, (b) r-chunk matching, d = ******0011******, (c) Hamming matching, $r = 8$, (d) R&T matching, $r = 0.5$.

The different matching rules and representations generate different types of detector covering shapes. This reflects the bias introduced by each representation and the matching scheme. It is clear that the relation of proximity exhibited by these matching rules in the binary self/non-self space does not coincide with the natural relation of proximity in a real-valued, two-dimensional space. Intuitively, this seems to make the task harder of placing these detectors to cover the non-self space without covering the self set. This fact is further investigated in the next section.

4 Comparing the Performance of Binary Matching Rules

This section shows the performance of the binary matching rules (as presented in section 2.1) in the NS algorithm. A generate-and-test NS algorithm is used. Experiments are performed using two synthetic data sets shown in Figure 3.

The first data set (Figure 3(a)) was created by generating random vectors (1000) in $[0, 1]^2$ with the center in (0.5,0.5) and scaling them to a norm less than 0.1, so that the points lies within a single circular cluster. The second set (Fig. 3(b)) was extracted from the Mackey-Glass time series data set, which has been used in different works that apply AIS to anomaly detection problems [7,14,13]. The original data set has four features extracted by a sliding window. We used only the first and the fourth feature. The data set is divided in two sets (training and testing), each one with 497 samples. The training set has only normal data, and the testing set has mixed normal and abnormal data.

4.1 Experiments with the First Data Set

Figure 4 shows a typical coverage of the non-self space corresponding to a set of detectors generated by the NS algorithm with r-contiguous matching for the first data set. The non-covered areas in the non-self space are known as *holes* [17] and are due to the

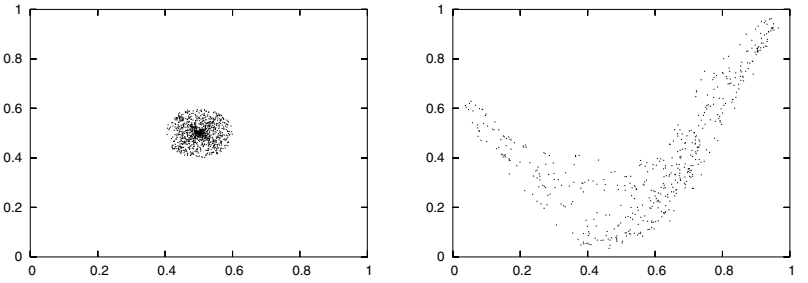

Fig. 3. Self data sets used as input to the NS algorithm shown in a two-dimensional real-valued problem space. (a) First data set composed of random points inside of a circle of radius 0.1 (b) Second data set corresponding to a section of the Mackey-Glass data set [7,14,13].

characteristics of *r*-contiguous matching. In some cases, these holes can be good: since they are expected to be close to self strings, the set of detectors will not detect small deviations from the self set, making the NS algorithm robust to noise. However, when we map the holes from the representation (self/non-self) space to the problem space, they are not necessarily close to the self set, as shown in Figure 4. This result is not surprising; as we saw in the previous section (section 3), the binary matching rules fail to capture the concept of *proximity* in this two-dimensional space.

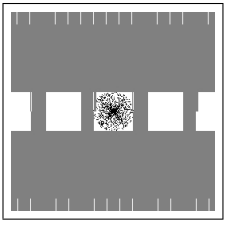

Fig. 4. Coverage of space by a set of detectors generated by NS algorithm using *r*-contiguous matching (with $r = 7$). Black dots represent self-set points, and gray regions represent areas covered by the generated detectors (4446).

We run the NS algorithm using different matching rules and varying the r value. Figure 5 shows the best coverage generated using standard (no Gray) binary representation. The improvement in the coverage generated by *r*-contiguous matching (Figure 5(a)) is due to the higher value of r ($r = 9$), which produces more specific detectors. The coverage with the *r*-chunk matching rule (Figure 5(b)) is more consistent with the shape of the self set because of the high specificity of *r*-chunk detectors. The outputs produced by the NS algorithm with Hamming and R&T matching rules are the same.

These two rules do not seem to do as well as the other matching rules (Figure 5(c)). However, by changing the encoding from binary to Gray (Figure 5(d)), the performance can be improved, since the Gray encoding changes the detector shape, as was shown in the previous section (Section 3). The change in the encoding scheme, however, does not affect the performance of the other rules for this particular data set.

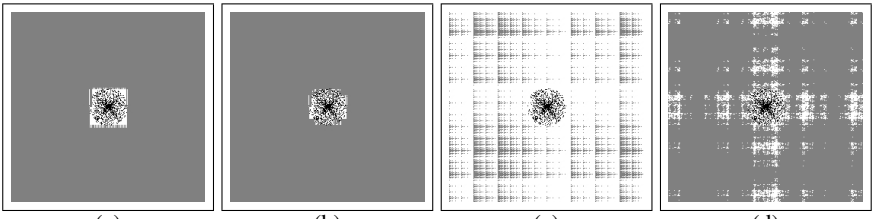

Fig. 5. Best space coverage by detectors generated with NS algorithm using different matching rules. Black dots represent self-set points, and gray regions represent areas covered by detectors. (a) r-contiguous matching, $r = 9$, binary encoding, 36,968 detectors. (b) r-chunk matching, $r = 10$, binary encoding, 6,069 detectors. (c) Hamming matching, $r = 12$, binary encoding (same as R&T matching, $r = 10/16$), 9 detectors. (d) Hamming matching, $r = 10$, Gray encoding (same as R&T matching, $r = 7/16$), 52 detectors.

The r-chunk matching rule produced the best performance in this data set, followed closely by the r-contiguous rule. This is due to the shape of the areas covered by r-chunk detectors which adapt very well to the simple structure of this self set, one localized, circular cluster of data points.

4.2 Experiments with the Second Data Set

The second data set has a more complex structure than the first one, where the data are spread in a certain pattern. The NS algorithm should be able to generalize the self set with incomplete data. The NS algorithm was run with different binary matching rules, with both encodings (binary and Gray), and varying the value parameter r (the different values are shown in Table 1). Figure 6 shows some of the best results produced. Clearly, the tested matching rules were not able to produce a good coverage of the non-self space. The r-chunk matching rule generated satisfactory coverage of the non-self space (Figure 6(b)); however, the self space was covered by some lines resulting in erroneously detecting the self as non-self (false alarms). The Hamming-based matching rules generated an even more stringent result (Figure 6(d)) that covers almost the entire self space. The parameter r, which works as a threshold, controls the detection sensitivity. A smaller value of r generates more general detectors (i.e. covering a larger area) and decreases the detection sensitivity. However, for a more complex self set, changing the value of r from 8 (Figure 6(b)) to 7 (Figure 6(c)) generates a coverage with many holes in the non-self area, and still with some portions of the self covered by detectors. So, this

problem is not with the setting of the correct value for *r*, but a fundamental limitation on of the binary representation that is not capable of capturing the semantics of the problem space. The performance of the Hamming-based matching rules is even worse; it produces a coverage that overlaps most of the self space (Figure 6(d)).

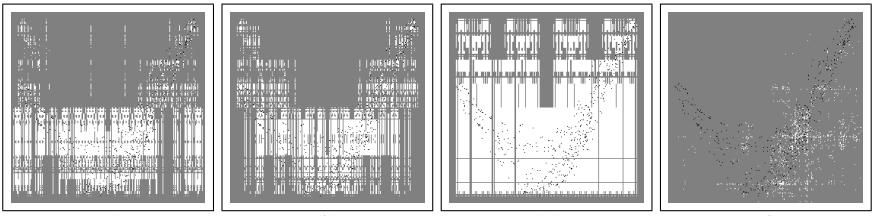

Fig. 6. Best coverage of the non-self space by detectors generated with negative selection. Different matching rules, parameter values and codings (binary and Gray) were tested. The number of detectors is reported in Table 1. (a) *r*-contiguous matching, $r = 9$, Gray encoding. (b) *r*-chunk matching, $r = 8$, Gray encoding. (c) *r*-chunk matching, $r = 7$, Gray encoding. (d) Hamming matching, $r = 13$, binary encoding (same as R&T matching, $r = 10/16$).

A better measure to determine the quality of the non-self space coverage with a set of detectors can be produced by matching the detectors against a test data set. The test data set is composed of both normal and abnormal elements as described in [13]. The results are measured in terms of the *detection rate* (percentage of abnormal elements correctly identified as abnormal) and the *false alarm rate* (percentage of the normal detectors wrongly identified as abnormal). An ideal set of detectors would have a detection rate close to 100%, while keeping a low false alarm rate. Table 1 accounts the results of experiments that combine different binary matching rules, different threshold or window size values (r), and two types of encoding. In general, the results are very poor. None of the configurations managed to deliver a good detection rate with a low false alarm rate. The best performance, which is far from good, is produced by the coverage depicted in Figure 6(b) (*r*-chunk matching, $r = 8$, Gray encoding), with a detection rate of 73.26% and a false alarm rate of 47.47%. These results are in contrast with other previously reported [7,21]; however, it is important to notice that in those experiments, the normal data in the test set is same to the normal data in the training set; so, no new normal data was presented during testing. In our case, the normal samples in the test data are, in general, different from those in the training set, though they are generated by the same process. Hence, the NS algorithm has to be able to generalize the structure of the self set in order to be able to classify correctly previously unseen normal patterns. But, is this a problem with the matching rule or a more general issue in the NS algorithm? In fact, the NS algorithm can perform very well on the same data set if the right matching rule is employed. We used a real value representation matching rule and followed the approach proposed in [14] on the second data set. The performance over the test data set

was detection rate, 94%, false alarm, 3.5%. These results are clearly superior to all the results reported in Table 1.

Table 1. Results of different matching rules in NS using the the second test data set. (r: threshold parameter, ND: number of detectors, D%: detection rate, FA%: false alarm rate). The results in bold correspond to the sets of detectors shown in Figure 6.

		Binary			**Gray**		
	r	**ND**	**D%**	**FA%**	**ND**	**D%**	**FA%**
***r*-contiguous**	7	0			40	3.96%	1.26%
	8	343	15.84%	16.84%	361	16.83%	16.67%
	9	4531	53.46%	48.48%	**4510**	**66.33%**	**48.23%**
	10	16287	90.09%	77.52%	16430	90.09%	75.0%
	11	32598	95.04%	89.64%	32609	98.01%	90.4%
***r*-chunk**	4	0			2	0.0%	0.75%
	5	4	0.0%	0.75%	8	0.0%	0.75%
	6	18	3.96%	4.04%	22	3.96%	2.52%
	7	98	14.85%	16.16%	**118**	**18.81%**	**13.13%**
	8	549	54.45%	48.98%	**594**	**73.26%**	**47.47%**
	9	1942	85.14%	72.97%	1959	88.11%	67.42%
	10	4807	98.01%	86.86%	4807	98.01%	86.86%
	11	9948	100%	92.92%	9948	100%	92.92%
	12	18348	100%	94.44%	18348	100%	94.44%
Hamming	12	1	0.99%	3.03%	7	10.89%	8.08%
	13	**2173**	**99%**	**91.16%**	3650	99.0%	91.66%
	14	29068	100%	95.2%	31166	100%	95.2%
Rogers & Tanimoto	9/16	1	0.99%	3.03%	7	10.89%	8.08%
	10/16	**2173**	**99%**	**91.16%**	3650	99%	91.66%
	11/16	29068	100%	95.2%	31166	100%	95.2%
	12/16	29068	100%	95.2%	31166	100%	95.2%

5 Conclusions

In this paper, we discussed different binary matching rules used in the negative selection (NS) algorithm. The primary applications of NS have been in the field of change (or anomaly) detection, where the detectors are generated in the complement space which can detect changes in data patterns. The main component of NS is the choice of a matching rule, which determines the similarity between two patterns in order to classify self/non-self (normal/abnormal) samples. There exists a number of matching rules and encoding schemes for the NS algorithm. This paper examines the properties (in terms of coverage and detection rate) of each binary matching rule for different encoding schemes.

Experimental results showed that the studied binary matching rules cannot produce a good generalization of the self space, which results in a poor coverage of the non-

self space. The reason is that the affinity relation implemented by the matching rule at the representation level (self/non-self) space cannot capture the affinity relationship at the problem space. This phenomenon is observed in our experiments with a simple real-valued two-dimensional problem space.

The main conclusion of this paper is that the matching rule for NS algorithm needs to be chosen in such a way that it accurately represents the data proximity in the problem space. Another factor to take into account is the type of application. For instance, in change detection applications (integrity of software or data files), where the complete knowledge of the self space is available, the generalization of the data may not be necessary. In contrast, in anomaly detection applications, like those in computer security where a normal behavior model needs to be build using available samples in a training set, it is crucial to count on matching rules that can capture the semantics of the problem space [4,20].

Other types of representation and detection schemes for the NS algorithm have been proposed by different researchers [4,13,15,21,23]; however, they have not been studied as extensively as binary schemes. The findings in this paper provide motivation to further explore matching rules for different representations. Particularly, our effort is directed to investigate methods to generate good sets of detectors in real valued spaces. This type of representation also opens the possibility to integrate NS with other AIS techniques like those inspired by the immune memory mechanism [9,22].

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). The authors would like to thank Leandro N. de Castro and the anonymous reviewers for their valuable corrections and suggestions to improve the quality of the paper.

References

1. J. Balthrop, F. Esponda, S. Forrest, and M. Glickman. Coverage and generalization in an artificial immune system. In *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 3–10, New York, 9-13 July 2002. Morgan Kaufmann Publishers.
2. J. Balthrop, S. Forrest, and M. R. Glickman. Revisting lisys: Parameters and normal behavior. In *Proceedings of the 2002 Congress on Evolutionary Computation CEC2002*, pages 1045–1050. IEEE Press, 2002.
3. C. A. C. Coello and N. C. Cortes. A parallel implementation of the artificial immune system to handle constraints in genetic algorithms: preliminary results. In *Proceedings of the 2002 Congress on Evolutionary Computation CEC2002*, pages 819–824, Honolulu, Hawaii, 2002.
4. D. Dagupta and F. González. An immunity-based technique to characterize intrusions in computer networks. *IEEE Transactions on Evolutionary Computation*, 6(3):281–291, June 2002.
5. D. Dasgupta. An overview of artificial immune systems and their applications. In D. Dasgupta, editor, *Artificial immune systems and their applications*, pages pp 3–23. Springer-Verlag, Inc., 1999.

6. D. Dasgupta and S. Forrest. Tool breakage detection in milling operations using a negative-selection algorithm. Technical Report CS95-5, Department of Computer Science, University of New Mexico, 1995.
7. D. Dasgupta and S. Forrest. Novelty detection in time series data using ideas from immunology. In *Proceedings of the International Conference on Intelligent Systems*, pages 82–87, June 1996.
8. L. N. de Castro and J. Timmis. *Artificial Immune Systems: A New Computational Approach*. Springer-Verlag, London, UK, 2002.
9. L. N. de Castro and F. J. Von Zuben. An evolutionary immune network for data clustering. *Brazilian Symposium on Artificial Neural Networks (IEEE SBRN'00)*, pages 84–89, 2000.
10. P. D'haeseleer, S. Forrest, and P. Helman. An immunological approach to change detection: algorithms, analysis and implications. In *Proceedings of the 1996 IEEE Symposium on Computer Security and Privacy*, pages 110–119, Oakland, CA, 1996.
11. J. D. Farmer, N. H. Packard, and A. S. Perelson. The immune system, adaptation, and machine learning. *Physica D*, 22:187–204, 1986.
12. S. Forrest, A. Perelson, L. Allen, and R. Cherukuri. Self-nonself discrimination in a computer. In *Proc. IEEE Symp. on Research in Security and Privacy*, pages 202–212, 1994.
13. F. González and D. Dagupta. Neuro-immune and self-organizing map approaches to anomaly detection: A comparison. In *Proceedings of the 1st International Conference on Artificial Immune Systems*, pages 203–211, Canterbury, UK, Sept. 2002.
14. F. González, D. Dasgupta, and R. Kozma. Combining negative selection and classification techniques for anomaly detection. In *Proceedings of the 2002 Congress on Evolutionary Computation CEC2002*, pages 705–710, Honolulu, HI, May 2002. IEEE.
15. 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.
16. S. Hofmeyr and S. Forrest. Architecture for an artificial immune system. *Evolutionary Computation*, 8(4):443–473, 2000.
17. S. A. Hofmeyr. An interpretative introduction to the immune system. In I. Cohen and L. Segel, editors, *Design principles for the immune system and other distributed autonomous systems*. Oxford University Press, 2000.
18. J. H. Holland, K. J. Holyoak, R. E. Nisbett, and P. R. Thagard. *Induction: Processes of Inference, Learning, and Discovery*. MIT Press, Cambridge, 1986.
19. N. K. Jerne. Towards a network theory of the immune system. *Ann. Immunol. (Inst. Pasteur)*, 125C:373–389, 1974.
20. J. Kim and P. Bentley. An evaluation of negative selection in an artificial immune system for network intrusion detection. In *GECCO 2001: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1330–1337, San Francisco, California, USA, 2001. Morgan Kaufmann.
21. S. Singh. Anomaly detection using negative selection based on the r-contiguous matching rule. In *Proceedings of the 1st International Conference on Artificial Immune Systems (ICARIS)*, pages 99–106, Canterbury, UK, sep 2002.
22. J. Timmis and M. J. Neal. A resource limited artificial immune system for data analysis. In *Research and development in intelligent systems XVII, proceedings of ES2000*, pages 19–32, Cambridge, UK, 2000.
23. P. D. Williams, K. P. Anchor, J. L. Bebo, G. H. Gunsch, and G. D. Lamont. CDIS: Towards a computer immune system for detecting network intrusions. *Lecture Notes in Computer Science*, 2212:117–133, 2001.