
Revisiting Negative Selection Algorithms

Zhou Ji

AutoZone, Inc., Memphis, TN 38103, USA

zhou.ji@ieee.org

Dipankar Dasgupta

Department of Computer Science, The University of Memphis, Memphis, TN 38152, USA

dasgupta@memphis.edu

Abstract

This paper reviews the progress of negative selection algorithms, an anomaly/change detection approach in Artificial Immune Systems (AIS). Following its initial model, we try to identify the fundamental characteristics of this family of algorithms and summarize their diversities. There exist various elements in this method, including data representation, coverage estimate, affinity measure, and matching rules, which are discussed for different variations. The various negative selection algorithms are categorized by different criteria as well. The relationship and possible combinations with other AIS or other machine learning methods are discussed. Prospective development and applicability of negative selection algorithms and their influence on related areas are then speculated based on the discussion.

Keywords

Artificial immune systems, negative selection algorithms, machine learning.

1 Introduction

Artificial Immune System (AIS) is an umbrella term that covers all the efforts to develop computational models inspired by biological immune systems. Among various mechanisms in the immune system that are explored for AIS, negative selection, immune network model, and clonal selection are still the most discussed models (Dasgupta, 1999; Dasgupta et al., 2003a; Garrett, 2005).

Discrimination between self and nonself is considered one of the major mechanisms in the complex immune system. Artificial negative selection is a computational imitation of self/nonself discrimination, first designed as a change detection method. It is modeled off the *T*-cell maturing process that happens in the thymus. *T*-cells of enormous diversity are first assembled with a pseudo-random genetic rearrangement process and those that recognize self cells are eliminated before the rest are deployed into the immune system to recognize and attack outside pathogens (Forrest et al., 1994; Dasgupta, 1999). The methods based on this process are generally called *negative selection algorithms* with the descriptive word “artificial” omitted. It is one of the earliest AIS algorithms to be applied in various real world applications. Since it was first conceived (Forrest et al., 1994), it has attracted many AIS researchers and practitioners and has gone through some phenomenal evolution.

While *negative selection algorithm* sometimes refers to this method in general, we usually call a specific algorithm based on this model a *negative selection algorithm* and use the plural form to refer to the entire family of such algorithms. Over the years, the name “negative selection algorithm” has been used by many very different models. In

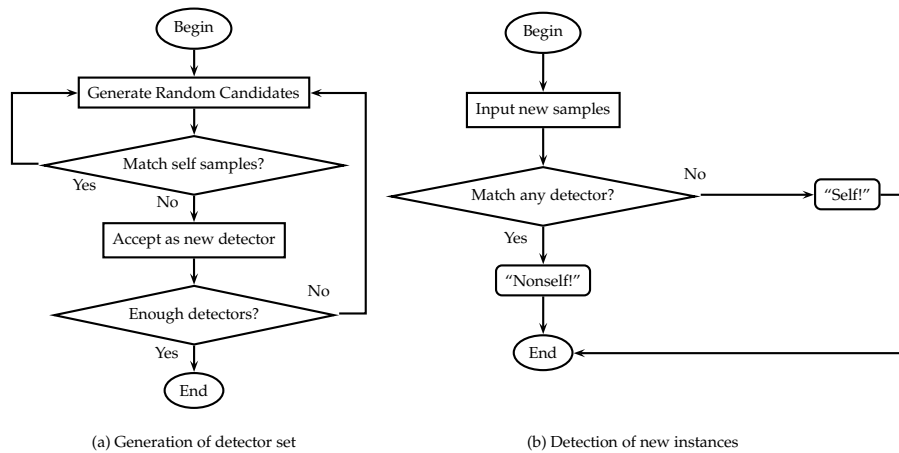


Figure 1: Outline of a typical negative selection algorithm.

spite of evolution and diversification of this method, the main characteristics of a negative selection algorithm described by Forrest et al. (1994) still remain. Figures 1 (a) and (b), similar to the original conception, describe the major steps in such an algorithm. In the generation stage, the detectors are generated by some random process and censored by trying to match self samples. Those candidates that match are eliminated and the rest are kept as detectors. In the detection stage, the collection of detectors (or detector set) is used to check whether an incoming data instance is self or nonself. If it matches any detector, it is claimed as nonself or an anomaly. This description is limited to some extent, but conveys the essential idea.

Despite the new models being proposed and the existing methods being improved continuously, the entire field of AIS including negative selection algorithms is still relatively young and not well defined. There always have been doubts about the applicability and uniqueness of negative selection algorithms. In the comprehensive survey on AIS by Garrett (2005), the author concluded that negative selection algorithms have a distinct process compared with other algorithms and may be most suitable for certain applications. On the other hand, Freitas and Timmis (2003) pointed out that negative selection algorithms are not appropriate to be used as a general classification method because it is one-class based. Stibor et al. (2005a,b) raised several possibilities to suggest that negative selection algorithms in general or certain specific models may not be appropriate, but the real weakness of the method, which likely may exist, was not satisfactorily identified. Both the relatively popular usage of this method in diverse applications and the doubt about its value justify a thorough review on this topic to facilitate its development or evolution. While other models in AIS are being studied actively and new ideas from immunology are pursued, negative selection algorithms deserve a more careful examination to avoid existing and potential misconceptions.

Deferring the discussion on variations, the most significant characteristics of a negative selection algorithm include:

1. Negative representation of information. From a machine learning point of view, we output the complementary concept of the real target concept. Although it is obviously distinct from any learning mechanism in positive space, its strength and applicability are still under exploration and controversy.

2. Using some form of detector set as the detection mechanism. This characteristic provides possibilities to extend this method to a distributed environment (Forrest et al., 1994; D'haeseleer et al., 1996), especially the chance to distribute the generation process.
3. One-class classification (Esponda et al., 2003a). Although the goal is to discriminate between two classes, samples from only one class (self or normal) are available to train the system. There are some works trying to generalize this method to classification of two classes (González et al., 2002) or multiple classes (Lee and Sim, 2004), mainly embedding a negative selection algorithm inside other algorithms. Nevertheless, what makes a negative selection algorithm unique is still the component that does one-class classification. This property casts doubt on its potential and its applicability as a classification algorithm (Freitas and Timmis, 2003).

The components in negative selection algorithms to be discussed in this review include data space representation, detector representation, matching rule, detector generation/elimination mechanism. Roughly speaking, each topic dominates the following ones in that order. For example, same data space representation is the basis on which we can discuss the difference of various detector representations, and so on. Other related issues include combination of different strategies (including coordination with dissimilar genre of algorithms, or with other AIS models, like positive selection), coverage analysis, and conceptual fundamentals, e.g. unification of various models (Esponda et al., 2003b), or the noteworthy concern on whether negative selection is a necessary and unique algorithm at all (Garrett, 2005).

Terminology in this area has hardly reached a consensus or a general convention. Very often the different phrases refer to the same idea; some researchers prefer immunological terms to mathematical/algorithmic terms while others like the opposite. It is thus necessary to reconcile some widely used terminology. In the following list, the terms in braces {} are the common alternatives referring to similar concepts as the leading words. The parentheses () indicate the immunological counterparts.

- detector (antibody)
- self samples {self set¹, training data} (self cells)
- incoming data instance {new data sample, data item}(antigen)
- distance measure {affinity measure, similarity measure} (affinity measure in the shape space)
- r -contiguous matching rule { r -contiguous bits rule, rcb matching rule}
- matching rule {match rule}

2 Variations and Techniques of Negative Selection Algorithms

2.1 Data Representation

Data representation is a fundamental difference between various models of negative selection algorithms. It limits the possible matching rules, the detector generation mechanism, and the detection process. Theoretically, binary representation subsumes any other representations because any data are eventually implemented as binary bits in a

¹Implying the assumption that self sample is *complete*. See §3.4.

computer. However, a matching rule defined on a high-level representation generally doesn't translate into a binary matching rule or rules in a straightforward way.

The common data to be processed include numeric data, categorical data, boolean data, and textual data. To deal with all these possibilities, most of the representation schemes can be grouped into two basic types: *string representation* and *real-valued vector representation*. In string representation, each data item, or a detector as a matter of fact, is represented as a string over a finite alphabet. The length of the string is usually fixed, but it is not necessarily always the case. Basically all four data types mentioned above can be processed using this representation. Binary representation, which is used in more works than other representations, is a special case of string representation. Despite the binary representation's low level role in a computer, most discussion on binary representation, at least in this context, equally applies to string representation over higher alphabet. On the other hand, numeric data, which has its intrinsic meaning of difference or similarity, cannot be processed properly if the matching rule doesn't reflect the distance in its numeric meaning. An earlier empirical study supported this conclusion (González et al., 2003a). Real-valued vector representation is thus another important type of representation. It represents each data item as a vector of real numbers. The matching rules and the measure of difference or similarity should be based on the numeric elements of the vector. It may be combined with string representation. For example, in hybrid representation, each data instance may consist of several features of different data types, e.g., integer, real value, categorical information, boolean value, text information, etc.

String representation's advantages are: 1) any data can be eventually represented in binary form; 2) it is easy to analyze; 3) it is good for textual or categorical information. Its limitations include the comprehensibility problem (difficult to interpret in the original problem space), the potential scalability issue (string size and matching threshold value), and some difficulty in combining with other techniques (conventional algorithms, machine learning, etc.) Negative selection algorithms using real-valued representation is often necessary even though theoretical analysis is not as convenient as in string representation.

On the next level, representation of data space doesn't define representation of detectors. In other words, representation of data space and representation of detectors do not need to be the same in general. The detector is usually described in the same data space, but more details may be involved unless the detector is purely a point. For example, if a detector is a hypersphere in n -dimensional real space, it can be represented by an n -dimensional point and a radius. If a detector is in fact a value range, it can be represented as a pair of values.

In real-valued vector representation, the self samples are seldom interpreted as the set of all the possible selves. This seems apparent in most real applications, but in many works using string representation, all the self patterns are assumed to be in the training data.

2.2 Matching Rules

A matching rule, which defines "matching" or "recognition", and the distance measure that the former is based on are the cornerstones in any detection, classification, or recognition algorithms, including negative selection algorithms. Matching rules are used both in the detector generation phase and in the anomaly detection phase. Regardless of representation, a matching rule M can be formally defined as

$$dMx \leftrightarrow \text{distance measure between } d \text{ and } x \text{ is within a threshold,}$$

where d is a detector and x is a data instance.

Matching threshold exposes the concept of partial matching: two points do not have to be exactly the same to be considered matching. Is a partial matching rule an approximation or a generalization? It is both. Generalization is the basic goal in many applications and it is often acceptable to have some approximation at the same time. In some cases, like most real-valued applications, the entire algorithm is meaningless without this generalization. On the other hand, it is sometimes necessary to model and analyze based on the complete self set to minimize approximation. In string representation, partial matching can be interpreted as extraction of rules or generalization of samples, but in real-valued vector representation, partial matching is almost always taken as generalization of self samples.

The choice of the matching rule or the threshold in a matching rule must be application specific and representation dependent, though some representations, matching rules, and control parameters may apply to more diverse applications.

2.2.1 Matching Rules in String Representation

Negative selection algorithms were first designed to detect change in strings, so string representation is a natural choice (Forrest et al., 1994). After an interesting preprocessing step to break strings of arbitrary length into shorter segments of predefined length, the *rcb* (***r*-contiguous bits**) matching rule is used. The rule was first proposed by Percus et al. (1993). Matching requirement is defined as r contiguous matching symbols in corresponding positions. Considering the fact that perfect matching is rare, the choice of *rcb* is mainly to simplify mathematical analysis with some flavor of immunology. In fact, such a partial match achieves generalization from limited self samples. That is an important characteristic of a learning algorithm. The value of r can be used to balance between more generalization or more specification. This rule has been the most popular one in the later works of negative selection algorithms.

Hamming distance or **edit distance** is another obvious choice of matching rule in string representation. The edit distance of two strings, s_1 and s_2 , is defined as the minimum number of point mutations required to change s_1 into s_2 , where a point mutation is to change a letter, to insert a letter, or to delete a letter. Edit distance, also called Levenshtein distance, is a generalization of Hamming distance. It can be further generalized by considering switching two letters as a single operation.

Balthrop et al. (2002) first proposed an ***r*-chunk matching rule**. It is a variant of the *rcb* matching rule and subsumes the former. Esponda et al. (2004) also claimed it as a simplification. An *r*-chunk detector is a string of r bits together with a specific *window*. The detector d is said to match a string x if all bits of d are equal to the r bits of x in the window specified by d . The difference from *rcb* boils down to the fact that the matching window is specified for each individual detector. A group of *r*-chunk detectors that cover all possible windows has the same effect as an *rcb* detector. An important difference between *rcb* and *r*-chunk matching rules is the holes, or the undetectable strings, that they may induce. Using full-length *rcb* matching rules, there are two types of holes: crossover holes and length-limit holes. A crossover hole is a "crossover" of certain self strings, in which all patterns of length r exist in self strings. A length-limit hole is one that has at least one window that does not exist in self strings and has some other windows that will match self strings. *r*-chunk matching rule eliminates the problem of length-limit holes. Because holes in a more general sense can be taken as part of the generalization, the *r*-chunk matching rule cannot be blindly judged better than *rcb* in this sense.

Permutation mask (Balthrop et al., 2002) plays an important role in matching rules like the *rcb* rule or *r*-chunk rule. If there is intrinsic order of the bits in the representation, specific bit order will hinder detection of some patterns because only the contiguous bits are considered to decide a match. For a different permutation mask, or equivalently different bit order in the representation, anomalies that otherwise cannot be detected could be caught. Whether and how the order of bits matters depend on specific applications. Hamming distance or edit distance, which is free from this problem, probably applies to more different applications. There is still interest in the role of permutation masks very lately (Stibor et al., 2006b).

Kim and Bentley (2001) reported difficulty in generating detector using the *rcb* rule with a low value of *r*. The overall detection rate becomes too low if *r* is chosen to be large enough to make generation of detector set feasible. However, Balthrop et al. (2002) reported a surprisingly good result for *r* as low as 1. The “magic bit” turned out to be application specific—the first bit is adequate to identify an internal IP. When there is no alternative analysis to identify such a special pattern, detectors with such low *r* appeared unreasonable. Kim and Bentley (2001) suggested an overall artificial immune model consisting of three different evolutionary stages: negative selection, clonal selection, and gene library evolution. Its detector is represented as a certain number of fields, some originally discrete and some discretized according to a value range. The final matching rule is in fact a variation of *rcb* where each ‘bit’ is a number. The significance of bit order introduced by *rcb* matching is not an inherent property of the original problem and hinders the success of the application.

Hofmeyr (1999) developed a refined theory of **multiple secondary representations** to reduce the number of trials to generate detectors on structured self by three orders of magnitude. The suggested secondary representations include pure permutation, imperfect hashing and substring hashing. The drawback is that all these still limit the matching at the level of genotype. They also add the cost of space and computing time.

González et al. (2003a) discussed the effect of different matching rules in binary representation in detail. A comparison was carried out to illustrate the difference between *rcb*, *r*-chunk matching, Hamming distance, and Rogers and Tanimoto (R&T) distance (a variation of Hamming distance) defined as

$$dMx \leftrightarrow \frac{\sum_i x_i \bar{\oplus} d_i}{\sum_i x_i \oplus d_i + 2 \sum_i x_i \oplus d_i} \geq r,$$

where \oplus is the exclusive-or operator, and $0 \leq r \leq 1$ is the threshold value. Numerical experiments are used to explore the behavior of these matching rules assuming the underlying problem space is a 2-dimensional real space $[0, 1]^2$, especially the shape of the area covered by individual detectors. Both the conventional binary representation and the Gray encoding, which maintains affinity at the binary level, were used. The general conclusion is that the binary presentation is not suitable to achieve generalization in such data space. That is not really surprising because similarity between two real values is not reflected in their binary representations. It is similar to the criticism by Freitas and Timmis (2003) of choosing a representation without considering a specific application.

Kaers et al. (2003) defined *antibody morphology* as the collection of basic properties of artificial antibody (or detectors) including shape, data-representation and data-ordering. It has much in common with the description of basic characteristics of a negative selection algorithm. Two new features are proposed: fuzzy antibody and Major Histocompatibility Complex (MHC), both oriented to arbitrary antibody morphology.

Assuming the data set S consists of a number data-strings of length l . Each data-string is denoted as (x_1, \dots, x_l) and all x_i are values belonging to attributes A_i . We associate a fuzzy member function (FMF) F_{ij} with it. Each F_{ij} converts a value x_i to a fuzzy membership value between 0 and 1. Attributes A_i can have different data types, FMFs, and matching thresholds. A detector is thus a string of FMFs and an r -contiguous symbols match rule M similar to the rcb matching rule can be used:

$$aMd \leftrightarrow \exists p, \forall q : p \leq q \leq p + r - 1 : F_{qj_q}(a_q) \geq th_q,$$

where th_q is the match threshold for attribute q . An immune response algorithm can take into account the match affinity f to measure how strong the match between an antibody and an antigen is. The proposed MHC metaphor is implemented as a permutation mask similar to Balthrop et al. (2002) and Dasgupta et al. (2003b) to handle the order of attributes.

The study by Harmer et al. (2002) is not mainly about negative selection, but its discussion of various matching rules between two strings is highly relevant here. It describes the following matching rules:

- Statistical correlation

The correlation coefficient produces a number between -1 and 1 that relates how similar the two strings are. It is defined as

$$\rho = \frac{\sum_{i=1}^N (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^N (X_i - \bar{X})^2 \sum_{i=1}^N (Y_i - \bar{Y})^2}},$$

where $X, Y \in \{0, \dots, 255\}^N$, $N = l/8$, l is the length of the binary string. The most common implementation uses ρ^2 to make it easier to compute.

- Binary distance

The most basic one is Hamming distance:

$$\text{Hamming similarity} = \sum_{i=1}^N \overline{(X_i \oplus Y_i)},$$

where $X, Y \in \{0, \dots, 1\}^N$.

Hamming distance was extended in several ways to produce the relative number of features that match or differ. These matching functions are all based on the following basic measures

$$a = \sum_{i=1}^N \zeta_i, \quad \zeta_i = \begin{cases} 1, & X_i = Y_i = 1 \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

$$b = \sum_{i=1}^N \zeta_i, \quad \zeta_i = \begin{cases} 1, & X_i = 1, Y_i = 0 \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

$$c = \sum_{i=1}^N \zeta_i, \quad \zeta_i = \begin{cases} 1, & X_i = 0, Y_i = 1 \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

$$d = \sum_{i=1}^N \zeta_i, \quad \zeta_i = \begin{cases} 1, & X_i = Y_i = 0 \\ 0, & \text{otherwise} \end{cases} \quad (4)$$

where $X, Y \in \{0, 1\}$. They are combined into different similarity functions:

1. Russel and Rao

$$f = \frac{a}{a + b + c + d}$$

2. Jacard and Needham

$$f = \frac{a}{a + b + c}$$

3. Kulzinski

$$f = \frac{a}{b + c + 1}$$

4. Sokal and Michener

$$f = \frac{a + d}{a + b + c + d}$$

5. Rogers and Tanimoto (the one discussed in González et al. (2003a))

$$f = \frac{a + d}{a + d + 2(b + c)}$$

6. Yule

$$f = \frac{ad - bc}{ad + bc}$$

r -contiguous bits matching belongs to this category too.

- Landscape-affinity matching

This was proposed to capture the ideas of matching biochemical, physical structure, and imperfect matching with a threshold of activation in an immune system.

The input string and the antibody strings are sampled as bytes and converted into positive integer values to generate a landscape. The two strings are then compared in a sliding window fashion. The comparison is made in one of the following three ways to produce an affinity measure and the measure is checked against a threshold.

1. difference matching rule

$$f_{\text{difference}} = \sum_{i=1}^N |(X_i - Y_i)|$$

2. slop-matching rule

$$f_{\text{slope}} = \sum_{i=1}^N |(X_{i+1} - X_i) - (Y_{i+1} - Y_i)|$$

3. physical matching

$$f_{\text{physical}} = \sum_{i=1}^N (X_i - Y_i) + 3 \times |\mu|,$$

where $\mu = \min(\forall i, (X_i - Y_i))$.

In summary, matching rules used in string representation include: rcb (r -contiguous bits), r -chunk, Hamming distance, edit distance, variations of Hamming distance (R&T distance, etc.), statistical correlation, and landscape matching.

2.2.2 Matching Rules in Real-Valued Vector Representation

The detectors in Dasgupta and González (2002) are represented as “detector rules” in a form like R^i : If $Cond_i$ then **nonself**, $i = 1, \dots, m$, where

$$Cond_i = X_1 \in [low_1^i, high_1^i] \text{ and } \dots \text{ and } X_n \in [low_n^i, high_n^i],$$

where m is the number of rules and n is the number of dimensions. A rule is considered good if it does not cover a positive sample (self samples) and its area is large. The condition part of each rule defines a hypercube in the descriptor space $([0, 1]^n)$. These condition parts are used as the individual chromosome in a genetic algorithm to generate detectors. Such detectors appear as a rectangular shape in the case of 2-dimensional data. The method accommodates the need for characterizing different levels of abnormality. A variability parameter v represents the level of allowed variability. While v itself is not part of the chromosome, different values of v are used to generate a hierarchical structure of rules grouped into different levels.

The matching rule proposed by González and Dasgupta (2003) is expressed as the membership function of the detector, which is a function of the detector-antigen distance and the radius of the detector

$$\mu_d(x) = e^{(-\|d-x\|^2)/(2r^2)}. \quad (5)$$

In other words, although the detector can be considered a hypersphere of the given center and radius, it is in fact a fuzzy sphere. The following distance measures were discussed:

- Euclidean distance

$$dist(s, K) = \|s - K\|$$

- Normalize distance: For a cluster, the standard deviation of the distance to the centroid, σ_K , describes the sparseness of the cluster. The distance is normalized based on σ_K .

$$dist(s, K) = \frac{\|s - K\|}{\sigma_K}$$

- D_∞ Minkowski distance (infinity norm distance): the maximum of the differences for all features:

$$dist(s, K) = \max \{|s - K|, \text{ for } i = 1, \dots, n\}$$

MILA (Multi-Level Learning Algorithm) (Dasgupta et al., 2003b) is a multi-level framework integrating both negative selections and positive selections. A unique feature in MILA is the *rcb* fashioned matching rule in real-valued representation. We can describe its distance measure as a *partial Euclidean distance*.

partial (Euclidean) distance: the distance defined over some of the elements of the vector. It is equivalent to the distance projected to a lower-dimensional space degraded from the original space.

In other words, the Euclidean distance used in a matching rule is not calculated over all the elements of the vector. Instead, only some elements are used to calculate Euclidean distance over a lower-dimensional space, similar to partial matching in string representation that only uses some bits. The elements included in the distance

measure can be chosen contiguously as in *rcb* rule or randomly. In both cases, the chosen positions need to match between the two points whose distance is calculated. The concept of permutation mask and crossover closure from string representation can be extended to the way these elements are chosen.

Branco et al. (2003) described a fault detection algorithm consisting of three high level modules: *T*-module, *B*-module and *D*-Module, mapping some components of a biological immune system. *T*-module is the self/nonself discrimination part, which basically is a negative selection algorithm. It uses real-valued vector representation and the matching rule is based on Euclidean distance. For each detector \vec{x} , if $|\vec{y} - \vec{x}| < r_x$, \vec{y} being the current state of system, not only does the detector report an anomaly, but it also describes the type of anomaly. Because of the second functionality, it is not desirable for detectors to overlap. When a candidate detector falls in the neighborhood domain of another detector or of a self sample, it is discarded and another candidate is generated at random. This results in detectors with no overlap. The *B*-module, which is an evolutionary system of vector population, sends an alert to *T*-module to adjust a detector's position or add a new detector when it identifies a cluster. *D*-module is a simplified *T*-module to facilitate distributing the monitoring systems throughout the network.

Another new variation of negative selection algorithms called *V-detector* was first developed in real-valued representation with a Euclidean distance-based matching rule (Ji and Dasgupta, 2004). Its detectors have individual sizes so the matching threshold is variable for the detector set. Furthermore, its strategy is orientated to treat distance and matching threshold in a generic way. For example, different distance measures can be plugged into the algorithm (Ji and Dasgupta, 2006).

Euclidean distance is not a panacea—for example, it is not desired when all the dimensions do not have equal weights in classification. The choice of distance measures still mainly relies on domain knowledge of a specific application. The distance used by Dasgupta et al. (2004) is Minkowski distance (λ -norm distance), defined as

$$D(x, y) = \left(\sum |x_i - y_i|^\lambda \right)^{1/\lambda}.$$

It is a generalization of Euclidean distance. Taylor and Corne (2003) compared the Euclidean distance matching rule and a *r*-bits matching rule using time series data processed with sliding windows. The raw real values are mapped to integers between 0 and 9. Even though the values used to calculate Euclidean distance were converted integer values, Euclidean distance still worked better. Based on the domain knowledge, they experimented with “differential encoding” to compare with the straight string representation. The results complicated the difference between two distance measures.

Hamaker and Boggess (2004) summarized several different distance measures, which apply to negative selection algorithms and are very useful to extend the models of the real-valued version, including to extend to heterogeneous data type.

- (normalized) Euclidean distance: It is not always the right choice simply because it is the first to come into your mind. The definition described by Hamaker and Boggess (2004) is divided by \sqrt{G} .

$$Euclidean(x, y) = \sqrt{\sum_{g=1}^G \left(\frac{x_g - y_g}{range_g} \right)^2}$$

- Manhattan: The version here is normalized as for Euclidean distance.

$$Manhattan(x, y) = \sum_{g=1}^G \frac{|x_g - y_g|}{range_g}$$

- Overlap: It is like Hamming distance. It only makes sense with a discrete or nominal attribute.

$$Overlap(x, y) = \sum_{g=1}^G overlap(x_g, y_g),$$

where

$$overlap(x_g, y_g) = \begin{cases} 0, & x_g = y_g \\ 1, & x_g \neq y_g \end{cases}$$

- Value Difference Metric (VDM): It is appropriate for symbolic data.

$$VDM(x, y) = \sum_{g=1}^G vdm(x_g, y_g) \cdot weight(x_g)$$

where

$$vdm(x_g, y_g) = \sum_{c=1}^C (P(c|x_g) - P(c|y_g))^2,$$

$$weight(x_g) = \sqrt{\sum_{c=1}^C P(c|x_g)^2}.$$

- Heterogeneous Euclidean-Overlap Metric (HEOM): It can handle mixed data with both continuous and nominal data.

$$HEOM(x, y) = \sqrt{\sum_{g=1}^G heom(x_g - y_g)^2},$$

where

$$heom(x_g, y_g) = \begin{cases} overlap(x_g, y_g), & g \text{ is nominal} \\ \frac{|x_g - y_g|}{range_g}, & g \text{ is discrete or real} \end{cases}$$

- Heterogeneous Value Difference Metric (HVDM): It is good for mixed data.

$$HVDM(x, y) = \sqrt{\sum_{g=1}^G hvdm(x_g - y_g)^2},$$

where

$$hvdm(x_g, y_g) = \begin{cases} \sqrt{vdm(x_g, y_g)}, & g \text{ is nominal} \\ \frac{|x_g - y_g|}{range_g}, & g \text{ is discrete or real} \end{cases}$$

- Discretized Value Difference Metric (DVDM): It allows for the use of VDM on real valued data.

$$DVDM(x, y) = \sum_{g=1}^G vmd(discretized(x_g), discretized(y_g)),$$

where

$$discretized(x_g) = \begin{cases} x_g, & x_g \text{ is discrete or nominal} \\ s - 1, & x_g = 1.0 \\ \lfloor x_g / width_g \rfloor, & \text{otherwise} \end{cases}$$

Matching rules are readily interpreted as detector shapes in real-valued representation. Straightforward Euclidean distance-based matching rule corresponds to hypersphere detectors. Shapiro et al. (2005) extended hypersphere detector to very general hyper-ellipsoid detectors. Such more flexibility could achieve better coverage of non-self space with fewer detectors, but the difficulty to generate such detectors considering more degrees of freedom has not been ideally solved. Hart and Ross (2004) and Hart (2005) discussed the shapes of the recognition-region in the context of idiotypic network model, but the same effect of detectors applies to negative selection algorithms as well.

In summary, major matching rules used in real-valued representation include: Euclidean distance, generalized distances of different norms in Euclidean space (including special cases: Manhattan distance (1-norm), Euclidean distance (2-norm), λ -norm distance for any λ , and infinity norm distance), interval-based matching, and other distance metrics.

2.3 Detector Generation Mechanism

2.3.1 Detector Generation with String Representation

The typical detector generation mechanism in negative selection algorithms, as described in the original model (Forrest et al., 1994), is a randomized algorithm that generates candidates and then eliminates those that match self samples or training data. Except for the difference in the matching rules developed later, most negative selection algorithms using string representation have the same or similar detector generation process. On the other hand, a few deterministic generation algorithms were also designed. In many cases, they were described so as to study analytically the algorithm complexity or detector coverage (D'haeseleer et al., 1996; Ayara et al., 2002; Wierzhon, 2000). Because string representations provide a more convenient platform for such analysis, deterministic algorithms are often discussed in such representations. D'haeseleer et al. (1996) discussed both randomized algorithms (exhaustive algorithm) and deterministic algorithms (linear time algorithm and greedy algorithm).

Kaers et al. (2003) categorized major detector generation algorithms into two types: those built heavily on the assumption of the string representation: linear, greedy, and binary template, and those relatively independent of *antibody morphology*: exhaustive, NSMutation. The exhaustive algorithm is the earliest one and follows closely the characteristics of natural negative selection process. In section 3.1, these algorithms and their complexity are discussed in more details.

Singh (2002) extended the greedy algorithm to higher alphabet. This is relevant in cases where the semantics of the information will be lost in binary representation. A lower number of false positives were reported compared with the results using binary

representation. It paralleled the works of generating all possible detectors by Stibor et al. (2004).

Hang and Dai (2004) introduced a new idea in detector generation by converting the data space into schemata space. Such a conversion compresses the data space. The problem space is a n -dimensional vector space including categorical features and numeric features. For real-valued features, a schema r is defined as the conjunction of the intervals as in the rules. Common schemata are those that are common in a group of rules. A number of common schemata are first evolved through a coevolutionary genetic algorithm in self-data space. The population used in the coevolutionary genetic algorithm consists of a number of non-interbreeding subpopulations. Species are initialized randomly, and new species are added into the population until the total number of species reaches a certain value. Then all the species are decoded into common schemata. Detectors are then constructed in the complementary space of the schemata using the traditional 'generate-and-test' strategy. The candidate detector (detection rule) is rejected if it contains any common schemata.

New algorithms of detector generation in string representation are still being proposed (Luo et al., 2006).

2.3.2 Detector Generation with Real-Valued Vector Representation

There has been, to date, no research reporting on the deterministic generation mechanism for real-valued vector representation. Theoretically, it is not impossible as long as the self region is defined in a way that the negative space can be represented explicitly, e.g., the case when self sample points are generalized to hyper-rectangular regions.

Randomized generation can be (1) the 'classical' generation-and-elimination strategy, (2) evolutionary approaches, e.g., genetic algorithm (Dasgupta and González, 2002), (3) one-shot randomized algorithm (Ji and Dasgupta, 2004), or (4) optimization with aftermath adjustment (Dasgupta et al., 2004).

Dasgupta and González (2002) used a genetic algorithm to generate detectors, which take the form of "detector rules". The fitness function is based on the volume of the rules (detectors), the number of self samples covered, and the overlap with other rules. A niching algorithm is applied to get the different rules. An asymmetric measure of the distance between the child and parent individuals is defined for deterministic crowding niching to give more importance to the area of the parent that is not covered. Although it doesn't really have a literally negative selection process, it is still claimed to be a negative selection algorithm partly because the negative representation of the target concept and the usage of a detector set. In comparison with a positive characterization method (*kd-tree*), this method appeared less precise but more efficient in time and space. Non-crisp characterization, in which the degree of abnormality is represented by a value within $[0, 1]$, was also introduced in this work.

González et al. (2003b) extended the earlier work on the real-valued negative selection algorithm (González et al., 2002) to a more sophisticated stage. The main development is (1) using Monte Carlo integration to estimate the volume of the self and nonself space, and hence the number of the detectors needed, and (2) simulated annealing to optimize the distribution of the detectors in the nonself space. The detectors in this case are basically hyperspheres of radius r in n -dimensional space. To estimate the total coverage of the detectors, an *effective covering volume* of a detector is defined as the volume of the inscribed hypercube, basically to compensate the possible overlap with other detectors. The volume of the entire self space is estimated with Monte Carlo integration assuming each self sample represents a self region of radius r_s . Based on those

estimates, the necessary number of the detectors can be decided before the generation process.

The initial detectors are generated randomly, which is likely to be very different from the optimal distribution assumed when doing the estimation. The detectors are then re-distributed with simulated annealing to approach the optimal distribution.

In another work by González and Dasgupta (2003), the mechanism used in the detector generation stage is not based on the same matching rule as in the detection stage. In the generation stage, the median distance is calculated between the randomly generated detector and k nearest neighbors. It is good to be noise-robust in some applications, but lacks consistency in the assumption for better analysis and comparison. If the distance is less than a threshold r , the detector will be moved away to the opposite direction of that distance; otherwise it is moved away from all other detectors by using the membership function defined in Equation (5). The size of moving step η is controlled as following

$$\eta_i \leftarrow \eta_0 e^{-i/\tau},$$

where η_0 is the initial value of adaption rate, and τ is a parameter that controls its decay. The stopping criterion is based on a pre-specified number of iterations. The simple stopping criterion in addition to the pre-set number of detectors implies that there is no direct control of coverage quality.

V-detector (Ji and Dasgupta, 2004, 2005), is a modified generate-and-test strategy. It liberates the algorithm from pre-deciding the number of the detectors by estimating coverage during the generation process. Its other unique feature is to maximize the sizes of individual detectors to achieve large coverage without much extra cost. It is also more concise than other real-valued negative selection algorithms and has the potential to extend to different representations. This method is efficient in: (1) to maximize the detectors does not add much cost because the censoring process needs to calculate the distances to all self samples any way; (2) coverage is estimated using the counter that is generated as a by-product of the attempts to add new detectors. Unlike other real-valued negative selection algorithms, overlap is generally not a concern in *V-detector*. The disadvantage is that (1) the number of detectors is not predictable though it is largely limited compared with similar methods; (2) if the detectors are ordered by their sizes, the larger detector doesn't always contribute more in coverage because of the overlap. That makes it hard to use the order to achieve flexible approximation by omitting small detectors.

Dasgupta et al. (2004) described another detector generation algorithm. An initial population of candidate detectors in hyperspherical shape are generated with random centers. Then, they mature through an iterative process. In each iteration, the radius of each detector is calculated as $r_d = D - r_s$, where r_s is a threshold value of self point. r_d is allowed to be negative in the current iteration until being moved or discarded in the next iteration. They are moved away from the training data and existing detectors. Then the detectors are evaluated by their size excluding the overlap, which is estimated roughly. The larger detectors are considered better fit and selected to go to the next generation. The smaller detectors are replaced with clones of those better-fit detectors. The clones are generated by moving the original detector by a fixed distance to its proximity. New detectors are also introduced to explore new areas of nonself space. The whole detector generation process terminates when a set of mature detectors are evolved.

2.4 Combining Negative Selection Algorithms with Other Methods

Dasgupta et al. (1999) conducted one of the earliest experiments combining positive selection with negative selection. The combined process is embedded in a genetic algorithm using a fitness function that assigns a weight to each bit based on the domain knowledge.

Ceong et al. (2003) proposed a method using a pair of complementary dual detector sets for two-class classification. Although it uses the typical negative detector idea, the different assumption is that we have samples (antigens) from both self and non-self classes. Basically both sets of detectors are based on negative selection, but using self samples and nonself samples to censor the candidates, respectively. Because of the initial negative selection process, neither detector set will make false positive error - implicitly assuming the self training data are complete. The chance of false negative is also largely reduced with the help of the complementary detector set.

Negative selection is combined with conventional classification algorithms by González et al. (2002). Negative selection is used to build a set of nonself samples first. Then, the original self samples and the generated nonself samples are fed together to classification algorithms. The detector generation strategy is a real-valued variation of the greedy algorithm in binary representation. If the detector matches the self sample, it is moved away to new location. It uses the average distance to k -nearest self samples instead of the distance to a single sample to decide matching so it is robust to noise and outliers. It is to some extent a strength, but more a difference in the assumption of self samples. Besides moving the detector candidate to the opposite direction of the centroid of k nearest self samples, it is also moved away from existing detectors so better coverage can be achieved. The approach doesn't depend on a specific type of classification algorithms. A multilayer neural network and an evolutionary algorithm were used, respectively, in the experiments. This work produces a fuzzy characterization of the normal space so a degree of normalcy is assigned to an element instead of giving a clear-cut judgment of normal or abnormal.

Kim and Bentley (2002) focused on immune memory using clonal selection model, but used detectors generated from negative selection to implement the memory mechanism.

González et al. (2005) combined a negative selection algorithm with self-organizing map (SOM). A negative selection algorithm was used to produce artificial anomalies required by a two-class method like SOM instead of detectors. This unique way of using negative selection algorithms lifts certain restrictions and makes them more useful. Hang and Dai (2005) use a similar strategy to synthesize an anomaly sample so as to facilitate usage of any learning algorithms that expect balanced samples to offer better performance.

3 Analysis of Negative Selection Algorithms

3.1 Complexity of Detector Generation

D'haeseleer et al. (1996) calls the original method "exhaustive detector generating algorithm", which finishes when the required number of detectors are generated; the number itself is chosen separately as a control parameter. The time complexity is

$$O\left(\frac{-\ln P_f}{P_m \cdot (1 - P_m)^{N_s}} \cdot N_s\right),$$

where P_m is the *matching probability*, the probability that a randomly chosen string and

a detector match, N_S is the size of self set. The space complexity is $O(l \cdot N_S)$. For specific matching rules, more conventional algorithms can be used. For example, D'haeseleer et al. (1996) proposed a self set-linear time algorithm for the *rcb* matching rule. Its time complexity is $O((1-r) \cdot N_S) + O((1-r) \cdot 2^r) + O(l \cdot N_R)$; space complexity is $O((1-r)^2 \cdot 2^r)$, where l is the string length; r is the number of contiguous bits in the matching rule. This is more costly in space than the exhaustive algorithm. Also depending on the *rcb* matching rule, the greedy algorithm (D'haeseleer et al., 1996) has higher time complexity $O((l-r) \cdot 2^r \cdot N_R)$ and the same space complexity as the above algorithm, but it provides the maximum coverage for a given number of detectors and the number of unmatched nonself strings is known.

Using a more traditional binary string and an *rcb* matching rule, Wierzchon (2000) described a deterministic algorithm to generate detectors that is more efficient in the sense of minimal number of detectors - or receptors as they are called in this work. Instead of generating a detector candidate randomly, it used a concept called *template*. A template is a string of length l over the alphabet $\{0, 1, *\}$. l is the original length of the strings in question. $*$ stands for "irrelevant", or "do not care". Each template $t_{i,w}$ has the substring of length k starting on position i that equals a binary string w of length k , and the remaining bits are all $*$ s. Note k has the same meaning as r in *rcb* matching rule. The set of all possible templates, T , thus contains $(l-k+1) \cdot 2^k$ different elements. T can be split into two disjoint subsets: T_S consisting of all the templates contained in at least one self string and T_N , the set of remaining templates that are used to construct detector (receptor) strings. Typically, T_S is a low fraction of T . T can be represented as a matrix that has 2^k rows, one for each different w , and $(l-k+1)$ columns, one for each starting point i . This notation makes it easier to analyze numerically. Given a detector r , the number of unique strings from U detected by r is

$$D(l, k) = 2^{l-k} + (l-k) \cdot 2^{l-k-1} = 2^{l-k-1} \cdot (2 + l - k).$$

This result can be extended from a binary alphabet to an alphabet of m symbols,

$$D_m(l, k) = m^{l-k} + (l-k) \cdot (m-1) \cdot 2^{l-k-1} = 2^{l-k-1} \cdot [(l-k) \cdot (m-1) + m].$$

The discriminative power of a whole set of detectors, however, is not the sum of all $D(l, k)$ s because different detectors can recognize common strings. Using a statistical approach, the average number of strings detected by n detectors is

$$d(l, k, n) = (1 - p_f(l, k, n)) \cdot 2^l,$$

where $p_f(l, k, n)$ is the failure probability

$$p_f(l, k, n) = (1 - p(l, k))^n \approx e^{-n \cdot p(l, k)},$$

where the approximation is valid for large n and small $p(l, k)$, the probability that two random strings match. For an "ideal" detector set, the coverage is fixed. For a given number of detectors that are not "ideal", we need to choose the detectors so that a maximum number of different templates are used. A binary tree is used to represent the connection between the self templates. Following the tree representation, we can reconstruct all possible "self strings" and find out those that are not self strings, namely, "holes". The holes, or the undetectables, can come from two sources: strings that can be built from templates in T_S , or self templates, and those that have nonself templates (templates in T_N). It is the former that are considered "holes" in this paper. This is similar to crossover closure in Esponda et al. (2003b).

Ayara et al. (2002) reviewed and compared five negative selection algorithms: Exhaustive, Linear, Greedy, Binary template, and NSMutation. The time and space complexities are examined for the five algorithms. Exhaustive detector generation is the original method proposed by Forrest et al. (1994). NSMutation is a version modified so guided mutation is performed instead of elimination. Compared with linear, greedy, and binary template algorithms that are highly restrictive to the *rcb* matching rule, NSMutation is more extensible. The greedy algorithm improved over the linear algorithm by eliminating the redundant detectors. The complexity of different detector generation algorithms is summarized in Table 1. Symbols used in this table are: N_S , number of self data; N_R , number of competent detectors; N_{R0} , number of candidates; r , matching threshold; m , alphabet size ($m = 2$ for binary representation); l , the string length. The time complexities of the exhaustive algorithm and NSMutation are exponential with respect to the size of self. All the others have linear time complexity. However, when the matching threshold r approaches length l , the linear complexities may behave similarly to that of the exhaustive and NSMutation algorithms due to the exponential value m^r in their time complexities. NSMutation has a higher space complexity than that of the exhaustive algorithm. The linear, greedy, and binary template algorithms all have higher space complexity, though the binary template has the lowest among them. The impact of guided mutation may only show up for an actually clustered self set. In addition, there is always a need to balance between the time taken to generate detectors and resultant good coverage of detectors (Ayara et al., 2002).

Algorithm	Time	Space
Exhaustive	$O(m^l \cdot N_S)$	$O(l \cdot N_S)$
Linear	$O((l - r + 1) \cdot N_S M^r) + O((l - r + 1) \cdot m^r) + O(l \cdot N_R)$	$O((l - r + 1)^2 \cdot m^r)$
Greedy	$O((l - r + 1) \cdot N_S M^r) + O((l - r + 1) \cdot m^r \cdot M_R)$	$O((l - r + 1)^2 \cdot m^r)$
Binary Template	$O(m^r \cdot N_S) + O((l - r + 1) \cdot m^r \cdot M_R)$	$O((l - r + 1) \cdot m^r) + O(N_R)$
NSMutation	$O(m^l \cdot N_S) + O(N_R \cdot m^r) + O(N_R)$	$O(l(N_S + N_R))$

Table 1: Complexity of different detector generation algorithm - string presentation.

The greedy algorithm extended to higher alphabet, proposed by Singh (2002), has a time complexity of $O(m^r \cdot (l - r) \cdot N_R)$, where N_R is the pre-specified number of detectors, and m is the size of alphabet. The space complexity is $O(m^r \cdot (l - r)^2)$.

Stibor et al. (2004) developed a deterministic algorithm to generate all possible detectors using the r -chunk matching rule. Such a detector set is called a *perfect detector set*, $D_{perfect}$, which contains a minimal number of detectors which recognize all elements in $U \setminus S$, where U is the representation space and S is the self set (assuming all self strings are included in S). The perfect detector set does not solve the problem of the *holes*. Instead, it clarifies that fact that holes are impossible to correct even when the complete self set and proper matching rule are given. The algorithm uses a hash table \mathcal{H} data structure to insert, delete and search efficiently boolean values, which are indexed with a composite key of r -chunk string concatenated with detector position p . It is divided into three phases. The total space size is $O(|\Sigma|^r)$. The time complexity of the entire three phases is $O((l - r) \cdot |\Sigma|^r) + O(|S| \cdot (l - r + 1)) + O(|\Sigma|^r) = O(|\Sigma|^r)$. The average number of generable detectors, depending on a self set S , r -chunk length

r , and alphabet size Σ was estimated as:

$$\left(1 - \frac{1}{(l-r+1) \cdot |\Sigma|^r}\right)^{|S| \cdot (l-r+1) \cdot (l-r+1)} \cdot (l-r+1) \cdot |\Sigma|^r.$$

3.2 Analysis of Detector Coverage

Negative selection algorithms are based on such an argument (Forrest et al., 1994) : given a reasonably large number of possible strings, the probability of two randomly chosen strings matching each other is relatively low; if the detector set is generated randomly and the abnormal strings can be considered as random to some extent, the probability that an abnormal string matches some detector increases with the number of detectors. Defining *failure probability* P_f as the probability that the detector set fails to detect a change, the main conclusion of Forrest et al. (1994) is

$$P_f \approx e^{-P_M N_R},$$

where P_M is the probability of a match between two random strings, and N_R is the number of detectors in the detector set. P_M is largely decided by three major control parameters: m , the number of alphabet symbols; l , the length of string; and r , the number of contiguous bits used in the matching rule. It was pointed out that the method worked better in some actual simulations than in the above analysis because the self samples, e.g., computer programs, are not just arbitrary strings. The question is how much of the nonself region is covered by the detector set. Estimating the coverage ratio, which is called *detector coverage*, or controlling it while the detectors are generated, is one of the major subjects of analysis for negative selection algorithms.

The greedy algorithm of detector generation described by D'haeseleer et al. (1996) guarantees the maximum possible coverage. Coverage is not only limited by imperfect detector set. For a given matching rule, there may exist "holes", the nonself region that cannot be covered by any valid detectors, because the self samples could have offsprings that are not in self set through crossover. It is hardly possible to have perfect coverage as for string representation, but a matching rule with variable threshold may possibly fill the holes better.

If a continuous data stream \hat{S} is being processed, preprocessing by splitting the stream into a set S of unordered strings of length l will lose some of the original information. D'haeseleer (1996) used the conditional entropy $H(\hat{S}|S)$ as a quantitative measurement of the loss of information.

$$\Delta I_1 = H(\hat{S}|S) = \log_2 \left(\frac{N_s}{N_1, N_2 \dots N_k} \right),$$

where N_s is the total number of strings in S ; k is the number of unique strings; $N_i, i = 1, \dots, k$, is the number of duplicates of unique string k . As a bound on the amount of information lost,

$$\Delta I_1 = O(N_s \cdot H(N_i/N_s)) = O(L_S \cdot H(N_i/N_s)/l),$$

where L_S is the number of bits in the stream \hat{S} . A smaller l , or an increased l when duplicates in S start to become fewer, will reduce the entropy or the loss of information and improve detection results. On the other hand, when the entropy is lower for smaller l , the search space will be bigger. We need to find a balance with the advantages of choosing proper length of string. D'haeseleer (1996) also formulated the second

source of information loss - negligence of the number of the duplicates. It is given as

$$\Delta I_2 = H(S|s_i) = \log_2 \left(\frac{N_s - 1}{k - 1} \right),$$

where s_i denotes one of the unique strings for $i = 1, 2, \dots, k$. Larger l will reduce ΔI_2 but may increase ΔI_1 in some cases, though it is small compared to ΔI_1 in general.

A *perfect detector set*, which recognizes all nonself strings that *can* be covered (excluding 'holes') is what we hope for. However, a perfect detector set could have to be of approximately the same size as the self set, which is not acceptable especially if the self set is large. By allowing a certain amount of error, a much smaller detector set may be enough. D'haeseleer (1996) gave an analysis of the number of detectors for a given failure probability, P_f , or the fraction of nonself strings that are not covered by the detector set. Denoting the information content (or entropy) of a self set S of size N_S as $H(S)$, and the information about S that is missing in the detector set R as $H(S|R)$, it is concluded that the difference between $H(S)$ and $H(S|R)$, called mutual information of S and R , is:

$$I(S; R) \equiv H(S) - H(S|R) \approx N_S \cdot \log_2(1/P_f).$$

For string length l and alphabet size m , we have the lower bound of detector size

$$N_R \geq \frac{N_S \cdot \log_2(1/P_f)}{l \cdot \log_2(m)}. \quad (6)$$

Another lower bound for N_R can be given in terms of matching probability P_m , the probability that a string and a detector that are randomly chosen match each other according to the specific matching rule is

$$N_R \geq (1 - P_f)/P_m.$$

Holes are unavoidable. For a partial matching rule, two self strings together may eliminate all detectors that are necessary to detect certain nonself strings. Given a string h and a matching rule M , if M has constant matching probability P_m , a self set of size $N_S = |M'(h)| = P_m \cdot N_U$, where $M'(h)$ is the set of the detectors matching string h , always suffices to induce holes.

Esponda et al. (2003b) developed a formal framework for analyzing positive and negative detection under various matching criteria. They discussed in depth the idea of crossover closure (Balthrop et al., 2002; Esponda et al., 2003a), which obviously has some root in genetic algorithm. The problem addressed is to derive some means for inductively determining the underlying concept from a collection of instances. The instances are represented as attribute vectors or strings. Most discussion applies to both positive and negative selection. The detection task can be interpreted as a *generation rule*. We have to emphasize that the term *generation* here has little to do with the generation of detectors. It is a rather confusing usage of the terminology from machine learning, referring to characterizing the underlying set (target concept) from a sample set.

Definition 3.1 A generation rule Q is a mapping from a set S of length l strings to a set $Q(S)$ of length l strings containing S .

Crossover closure was first based on the contiguous *windows* of attributes and then extended to any subset of features. Formally, given a set S of strings, and a fixed $1 \leq$

$r \leq l$, the crossover $CC(S)$ of S is defined in terms of its features as

$$CC(S) = \{u \in U | (\forall \text{ features } w) (\exists s \in S) u[w] = s[w]\},$$

where U is the set of all possible strings and $u[w]$ is the projection of string u onto feature w . Features can be interpreted of some subset of the attributes describing the data items. The most common choice of such a subset is r contiguous bits, in which case $l - r + 1$ overlapping r -chunk matching detectors can be obtained by decomposing a rcb detector. In general, a feature can even be any combination or function of attributes of the instance vectors. The crossover closure is a proper subset of the possible strings generated by the GA's crossover operator.

An r -chunk matching rule can be viewed as a generalization of the rcb matching rule. It is highly related with the concept of crossover closure. r -chunk matching rules and rcb matching rules are not equivalent in terms of the languages they recognize, but we can decompose an rcb matching rule into overlapping r -chunks as mentioned above. If the set of self instances S is a random, uniformly generated collection of strings defined over some finite alphabet A , the number of negative detectors is given by

$$E_{neg} = t(\mathbb{A}^r - E_r),$$

where $E_r \approx \mathbb{A}^r - \mathbb{A}^r(1 - \mathbb{A}^{-r})^{|S|}$, $t = l - r + 1$ is the number of windows.

A similar conclusion has been described for positive selection too. To compare the strength of negative detection versus positive selection, let us compare the number of detectors. The numbers of detectors for the two strategies are equal when

$$|S| \approx 0.693\mathbb{A}^r.$$

3.3 Detection Performance

The overall performance of negative selection algorithms depends on the complexity of detector generation (see §3.1) and the complexity of checking new data instances. In most models, checking a new instance takes linear time with respect to the number of detectors. From both aspects of generation and detection, the method would be worthless if the detector set is too large to be better than other straightforward detection mechanism, for example, directly checking self samples.

Performance under dynamically changing normal behavior is another issue. Forrest et al. (1994) indicated that this may be the weakness of the method considering the possible need to re-generate the entire detector collection when the self set alters. In the context of change detection, the addition of new features are readily detected by the existing detector set, but the deletion of features will not be detected.

3.4 Assumptions in Comparison

As for the purpose of research on the methodology, the data set is ideally collected under full control and in a realistic context. In real applications, we cannot expect data to be exactly in certain way as the method requires. Nevertheless, assumptions about the data and the problem were often not properly clarified in evaluation and comparison of negative selection algorithms.

The following issues concerning the type of the problems and the property of the training data are important in any analysis of negative selection algorithms:

- frequency-reflecting data: The distribution probability of data is crucial to evaluate the successfulness of the learning algorithm.

- noisy data: Is there a need of mechanism to deal with noise and outliers, or any other “bad” self samples? In real applications, it seems necessary in most scenarios. However, it is more illustrative to separate the issue for many applications.
- recognizing each abnormal data or reacting only when the same anomaly happens multiple times. The later is more reasonable in the application like network intrusion detection or fault detection.
- completeness of the self samples. If all self patterns are represented in the training data, we call it “complete”. It is usually not the case for real-valued representation, but it is true for many applications using string representation.
- dynamic data: This refers to the training data that are not stationary (self set changes over time).
- distributed data: In this case, the self set is distributed or too large to observe completely.

Forrest et al. (1994) mentioned that the method relies on the fact the data are not corrupted when the detectors are generated. This reflects the idea that the self samples are at least considered correct regardless of whether they are complete or not. Even if the self samples are complete as well as correct, negative selection algorithms are still probabilistic in most methods, implying that they may not achieve perfect coverage. The goal is to have a small number of detectors that are capable of detecting a relatively large portion of nonself space. In short, the algorithms should depend on the data properties as little as possible, but the common assumptions are important for a plausible comparison.

4 Exploring the Fundamentals of Negative Selection Algorithms

4.1 Effort to Establish a Framework

Esponda et al. (2003b) approached, in a very generic way, any anomaly detection problems in which the model of normal behavior is constructed from an observed sample of normally occurring patterns. Such a model represented either the set of allowed patterns (positive detection) or the set of anomalous patterns (negative detection/selection). Real-valued representation doesn't fit into this framework. Otherwise, the framework is rather complete in terms of covering various variation of negative selection algorithms. Three generation rules are presented: Hamming radius, crossover closure, and n -gram. The n -gram matching rule is different from crossover in two ways: first, the restriction of fixed string length l is relaxed; second, the information of window location is ignored. If we consider the n -gram length n as the string length, this matching rule is in fact reduced to perfect matching although we need to preprocess original string input into strings of length n with a sliding window. The taxonomy of detection schemes established consists of three dimensions: the form of a single detector together with matching rule; positive versus negative selection; disjunctive versus conjunctive matching. The class of languages recognized by negative disjunctive selection (the most common flavor of negative selection) is the same as that of positive conjunctive detection under rcb or r -chunk matching rules. The crossover closure of a sample S exactly characterizes the class of languages recognized by negative disjunctive selection.

Esponda et al. (2003b) pointed out that the normal and anomalous sets both contain the same amount of information, which suggests that there might be equally compact representation. The novel concept of a negative database was proposed consequently (Esponda et al., 2004).

Hofmeyr and Forrest (2000) extended the intrusion detection system LISYS (Balthrop et al., 2002) to the architecture ARTIS (ARTificial Immune System) that may apply to various domains. Nevertheless, it is obviously oriented to the application of intrusion detection on the network. A negative selection algorithm is the main component in the model. It is extended to dynamic application in which detector sets are updated according to an updated connotation of normal behavior. Detectors can die not only due to negative selection of new self samples, but also through old age or lack of co-stimulation.

Activation threshold, sensitivity levels, and co-stimulation are described to reduce false positives. Two methods are implemented to deal with the issue of false positive that happens when we have an “incomplete self set”, meaning that generalization beyond the training set is necessary for some strings that are not in a training set to be taken as legitimate self string. The first method is *activation threshold*: only when a detector matches multiple times, say, more than n times, is anomaly declared. Note this happens at detection stage instead of detector generation time. Even when a new string is “detected”, we do not claim an intrusion detected until the number of occurrences exceeds the threshold. Second, *adaptive activation* is used to decrease the activation threshold by one whenever a detector is activated. This “sensitizing” behavior is to some extent like inflammation or other mechanisms that increase the sensitivity of natural immune system. Activation threshold added another layer to the model of a general negative selection. It applies to the situation when we are not trying to capture individual abnormal data items. A network packet is a typical example of such an application.

Affinity maturation, resembling a genetic algorithm, is included so in the case where two detectors match, the one with closer match wins and the other one is ignored. For *rcb* or *r*-chunk matching rules, closer matching can be interpreted as matching of more bits, though “closer match” was not explicitly defined in the original work. A *second signal* or *co-stimulation* is used to curb false positive. It is implemented as the email from the human reviewer. The email is sent to confirm a true positive case. That type of human confirmation mechanism is used in some other AIS models too.

Hofmeyr and Forrest (2000) also pointed out that negative selection algorithms resemble the architecture of a classifier system from several aspects, including comparison with Holland’s classifier system.

Some effort was also made to find the link between negative selection algorithms and some mathematical formality with the hope of better analysis (Stibor et al., 2006a).

Balachandran et al. (2007) described a framework for multi-shaped detectors in the real-valued representation.

4.2 Uniqueness and Applicability

Freitas and Timmis (2003) highlighted the issue of the potential pitfalls of bias in the selection of data representation and the use of various affinity measures. They also argued that negative selection is not appropriate for classification. A classification algorithm may have inductive bias of two major types: representation bias and preference bias. In the context of negative selection, the former is related to the bias introduced by the specific data representation. The latter is directly related with the distance measures in a matching rule. It is pointed out that adapting data to the algorithm will throw

away relevant information and limit the algorithm's effectiveness. It is acclaimed that the affinity measure should fit the data type. For binary representation, Hamming distance is the simplest and the most natural. The authors disagree that the *rcb* matching rule are chosen because it is "biologically plausible". For real-valued representation, both Euclidean and Manhattan distances seem natural choices although the latter were less used. This work emphasizes that the choice should be made based on the data being mined instead of arbitrarily. For example, categorical data is very common in real world data but under-represented in the research concerning affinity measure.

Garrett (2005) assessed all the existing major types of AIS, including negative selection, in terms of distinctiveness and effectiveness. The paper tried to answer one of the most fundamental questions in the area: whether the concept of AIS and its individual types are useful or necessary at all. It was concluded that negative selection is not unique in symbol and expression compared with other methods including genetic and evolutionary computation and artificial neural networks, but its process is unique and the results are unique as well. It is not conclusive about the quality of the results as compared with existing methods. Whether negative selection works faster than other method cannot be confirmed either.

The negative database (Esponda et al., 2004) is the one of latest development in negative selection algorithms, which has raised another potential advantage of negative selection algorithms: hiding the descriptive information of self while providing a way to identify nonself data. Given a set of fixed strings, called positive database DB (*self*), all the possible records or strings not in DB constitute the notion of negative database $NDB = U - DB$ (*nonself*), where U denotes the universe of all possible strings of the same length defined over the same alphabet. Binary string presentation was used to facilitate the discussion. The negative database is then represented using alphabet $\{0, 1, *\}$, where $*$ is the 'don't care' symbol. The interesting property of this representation concerns the difficulty of inferring DB from given NDB . For an arbitrary set of strings defined over $\{0, 1, *\}$, determining which strings are not represented in NDB is an \mathcal{NP} -hard problem. Basic database operations like initialization, insertion and deletion were described for the negative database. The proposed algorithms for these operations may cause the size of NDB to grow unreasonably. It is important for any implementation to control the number of entries that match a particular string.

Ebner et al. (2002) suggested that the negative selection algorithms not copy the natural negative selection too closely. The use of negative selection is not a reasonable choice if space is finite and self comprises only a small fraction of the available space or if space is infinite. Negative selection algorithms, at least in their original form, distribute detectors randomly over an n -dimensional space. If the space is infinite or the space is finite but self comprises only a small fraction of the total space, it makes more sense to describe the self directly. User authentication using keystroke analysis is used as an example to illustrate the idea. It is one of typical cases where negative selection algorithms did not work when they were not used properly.

The work by Lee and Sim (2004) is an example of using negative selection as a classification algorithm that was questioned by Freitas and Timmis (2003). Its matching rules are based on Hamming distance of fixed-length substrings. Two sizes of alphabet are used for classification at two different levels. One set of detectors is generated for each pattern (class). Confirmation to a given pattern is concluded when the input is rejected by all other detector sets.

Ji and Dasgupta (2006) discussed inconsistency in the terminology and the assumptions used in negative selection algorithms to clarify the confusion or misunder-

standing in applicability. The main concern is whether negative selection algorithms can be used in certain problems, or in any problem at all. The limitations of specific representation or of certain basic assumptions, e.g. one-class training, are differentiated from the limitation of negative selection algorithms in general. Confusing concepts, for example, the detection rate and detector coverage, are also discussed.

4.3 Comparison with Other Methods

Comparison of negative selection algorithms' performance with other methods is still rare due to the fact that the basic assumption is different, e.g., negative selection's advantage of training with normal samples only.

Dasgupta and Forrest (1999) compared negative selection with an artificial neural network approach called ART. The binary version of ART, ART1, was used in the comparison. An ART network detects anomaly deterministically. The vigilance threshold ρ controls sensitivity as matching threshold r does in negative selection algorithms. Negative selection algorithms recognize a novel pattern in a decentralized way while ART makes a global decision in recognizing an input.

Hofmeyr and Forrest (2000) lined out more detailed comparison with a traditional classifier system. It collated major components in a learning classifier system with their counterparts in the negative selection system ARTIS. Some of them play very similar roles. Others are different in their functionality or implementation. ARTIS also has some unique features, e.g., multiple representations, which do not exist in a classifier system.

González and Dasgupta (2003) compared negative selection algorithms and self-organizing map (SOM), a type of neural network that captures the features in the input and provides a structural representation.

5 Summary

This review focuses on revisiting various negative selection algorithms in a survey form to provide up-to-date information to the newcomers in the field. There are many reports of successful applications of negative selection algorithms. However, currently the theoretical foundation of this method is still developing in the sense that most applications have their own versions of the algorithms, including variability in the data representation, detector representation, detector generation mechanism, etc. A widely accepted standard model or taxonomy is still to be established for this method to mature. Development of common terminology and model taxonomy is also necessary for more enlightening analysis.

The negative selection algorithms' uniqueness and strength are the basis for the entire discussion. Its advantage can be grouped into two levels. The fundamental level includes some features that make this method really special:

- No prior knowledge of nonself is required (D'haeseleer, 1996).
- It is inherently distributable; no communication between detectors is needed (D'haeseleer et al., 1996).
- It can hide the self concept (Esponda et al., 2004).

At the other level, it has various strengths that may not be totally unique to this method:

- Compared with other change detection methods, negative selection algorithms do not depend on the knowledge of defined "normal". Consequently, checking activ-

Author and year	New features/conclusions	Data representation	Matching rules
Forrest et al. (1994)	negative selection first proposed	string	<i>rcb</i>
D'haeseleer et al. (1996)	analysis of information loss	string	<i>rcb</i>
Dasgupta and Forrest (1999)	time series data with sliding windows	string	<i>rcb</i>
Dasgupta et al. (1999)	multiple classes, combined with positive selection	string	weighted bit matching
Hofmeyr and Forrest (2000)	activation threshold, life cycle of detectors, costimulation, etc.	string (49-bit binary)	<i>rcb</i>
Wierzbach (2000)	deterministic approaches to decide the number of strings detected and to generate detectors	binary string	<i>rcb</i>
Kim and Bentley (2001)	unique representation of network traffic data	real value-string	<i>rcb</i>
González et al. (2002)	combined with classifier	real vector	Euclidean distance to centroid
Balthrop et al. (2002)	<i>r</i> -chunk matching rule	string (49-bit binary)	<i>r</i> -chunk
Esponda et al. (2003b)	taxonomy of detection scheme, analysis on number of detectors	string	<i>rcb</i> , <i>r</i> -chunk, <i>n</i> -gram, Hamming distance
Ayara et al. (2002)	NSMutation detector generation	string	<i>rcb</i>
Dasgupta et al. (2003b)	multi-level, various components	real vector	partial Euclidean distance
Branco et al. (2003)	additional alerting mechanism to complement the detector set	real vector	Euclidean distance
Ji and Dasgupta (2004)	maximized detectors	real vector	Euclidean distance
Esponda et al. (2004)	negative database	string representation	<i>r</i> -chunk
Ji and Dasgupta (2005)	integrated coverage estimate in detector generation	real vector	Euclidean distance

Table 2: Timeline of negative selection algorithms

ity of each site can be based on a unique signature of each while the same algorithm is used over multiple sites.

- The quality of the check can be traded off against the cost of performing a check (Forrest et al., 1994).
- Symmetric protection is provided so the malicious manipulation on detector set can be detected by normal behavior of the system (Forrest et al., 1994).
- If the process of generating detectors is costly, it can be distributed to multiple sites because of its inherent parallel characteristics.
- Detection is tunable to balance between coverage (matching probability) and the number of detectors (D'haeseleer, 1996).

Negative selection algorithms played an important role in the research of artificial immune systems. They are still actively used in various applications although there are now more new frontlines in AIS research. Table 2 shows the timeline of this method's development.

Data space representation is the basic difference between various negative selection algorithms. As the key components of negative selection algorithms, matching rules and the highly related detector representation vary in different data representations although some conceptual overlap does exist. Distance measure is an important part of matching rules and needs more systematic experiments and comparisons to understand its influence. Compared with binary or string representation, formal analysis is more needed for the genre of real-valued representation.

Although negative selection algorithms are widely accepted as a specific type of AIS, they in fact include many algorithms or models that are very different from each other. Other than a few core characteristics, some important elements in these algorithms, e.g. the detector generation process, could be totally different. General judgment on negative selection algorithms could be misleading if such variety is ignored.

References

- Ayara, M., Timmis, J., de Lemos, R., de Castro, L., and Duncan, R. (2002). Negative selection: How to generate detectors. In Timmis, J. and Bentley, P. J., editors, *Proceedings of the 1st International Conference on Artificial Immune Systems (ICARIS)*, volume 1 of 89-98, University of Kent at Canterbury.
- Balachandran, S., Dasgupta, D., Nino, F., and Garrett, D. (2007). A framework for evolving multi-shaped detectors in negative selection. In *Proceedings of IEEE Symposium Series on Computational Intelligence*, Honolulu. IEEE Press.
- Balthrop, J., Esponda, F., Forrest, S., and Glickman, M. (2002). Coverage and generalization in an artificial immune system. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2002)*, pages 3–10, New York. Morgan Kaufmann Publishers.
- Branco, P. J. C., Dente, J. A., and Mendes, R. V. (2003). Using immunology principle for fault detection. *IEEE Transactions on Industrial Electronics*, 50(2):362–373.
- Ceong, H. T., Kim, Y.-I., Lee, D., and Lee, K.-H. (2003). Complementary dual detectors for effective classification. In J. Timmis, P. Bentley, and E. Hart (Eds), *Proceedings of Second International Conference on Artificial Immune System (ICARIS 2003)*, pages 242–248. Springer.
- Dasgupta, D. (1999). An overview of artificial immune systems and their applications. In Dasgupta, D., editor, *Artificial Immune System and Their Applications*, pages 3–23. Springer-Verlag.

- Dasgupta, D., Cao, Y., and Yang, C. (1999). An immunogenetic approach to spectra recognition. In Banzhaf, W., Daida, J., Eiben, A. E., Garzon, M. H., Honavar, V., Jakiela, M., and Smith, R. E., editors, *Proceedings of Genetic and Evolutionary Computational Conference (GECCO 1999)*, volume 1, pages 149–155, Orlando, Florida. Morgan Kaufmann.
- Dasgupta, D. and Forrest, S. (1999). An anomaly detection algorithm inspired by the immune system. In Dasgupta, D., editor, *Artificial Immune System and Their Applications*, pages 262–277. Springer-Verlag.
- Dasgupta, D. and González, F. (2002). An immunity-based technique to characterize intrusion in computer networks. *IEEE Transactions on Evolutionary Computation*, 6(3):1081–1088.
- Dasgupta, D., Ji, Z., and Gonzalez, F. (2003a). Artificial immune system (AIS) research in the last five years. In R. Sarker, R Reynolds, Hu. Abbass, K. Chen Tan, B. McKay, D. Essam, and Tom Gedeon (Eds), *Proceedings of The 2003 Congress on Evolutionary Computation (CEC 2003)*, pages 123–130, Canberra, Australia. IEEE Press.
- Dasgupta, D., KrishnaKumar, K., Wong, D., and Berry, M. (2004). Negative selection algorithm for aircraft fault detection. In G. Nicosia et al. (Eds), *Proceedings of Third International Conference on Artificial Immune Systems (ICARIS 2004)*, pages 1–13. Springer.
- Dasgupta, D., Yu, S., and Majumdar, N. S. (2003b). MILA - multilevel immune learning algorithm. In E. Cantu-Paz et al. (Eds), *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2003)*, LNCS 2723, pages 183–194, Chicago, IL. Springer.
- D’haeseleer, P. (1996). An immunological approach to change detection: Theoretical results. In *Proceedings of the 9th IEEE Computer Security Foundations Workshop*, pages 18–27. IEEE Computer Society Press.
- D’haeseleer, P., Forrest, S., and Helman, P. (1996). 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. IEEE Computer Society Press.
- Ebner, M., Breunig, H.-G., and Albert, J. (2002). On the use of negative selection in an artificial immune system (MPP). In E. Cantu-paz et al. (Eds), *Proceeding of the International Conference Genetic and Evolutionary Computation (GECCO 2002)*, pages 957–964, New York. Morgan Kaufmann.
- Esponda, F., Ackley, E. S., Forrest, S., and Helman, P. (2004). Online negative databases. In G. Nicosia, V. cutello, P. J. Bentley, and J. Timmis (Eds), *Proceedings of Third International Conference on Artificial Immune Systems (ICARIS 2004)*, pages 175–188. Springer.
- Esponda, F., Forrest, S., and Helman, P. (2003a). The crossover closure and partial match detection. In J. Timmis, P. Bentley, and E. Hart (Eds), *Proceedings of Second International Conference on Artificial Immune System (ICARIS 2003)*, pages 249–260, Edinburgh, UK. Napier University. Springer.
- Esponda, F., Forrest, S., and Helman, P. (2003b). A formal framework for positive and negative detection schemes, pages 357–373. IEEE Systems, Man, and Cybernetics Society.
- Forrest, S., Perelson, A., Allen, L., R., and Cherukuri (1994). Self-nonsel self discrimination in a computer. In *Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy*, pages 202–212, Los Alamitos, CA. IEEE Computer Society Press.
- Freitas, A. A. and Timmis, J. (2003). Revisiting the foundation of artificial immune systems: A problem-oriented perspective. In J. Timmis, P. Bentley, and E. Hart (Eds), *Proceedings of Second International Conference on Artificial Immune System (ICARIS 2003)*, pages 249–260. Springer.
- Garrett, S. M. (2005). How do we evaluate artificial immune systems? *Evolutionary Computation*, 13(2):145–178.

- González, F., Dasgupta, D., and Gómez, J. (2003a). The effect of binary matching rules in negative selection. In E. Cantu-Paz et al. (Eds), *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2003)*, LNCS 2723, pages 195–206, Chicago, IL. Springer.
- González, F., Dasgupta, D., and Kozma, R. (2002). Combining negative selection and classification techniques for anomaly detection. In *Congress on Evolutionary Computation (CEC 2002)*, pages 261–272, Hawaii. IEEE Press.
- González, F., Dasgupta, D., and Nino, L. F. (2003b). A randomized real-value negative selection algorithm. In J. Timmis, P. Bentley, and E. Hart (Eds), *Proceedings of Second International Conference on Artificial Immune System (ICARIS 2003)*. Springer.
- González, F. A. and Dasgupta, D. (2003). Anomaly detection using real-valued negative selection. *Genetic Programming and Evolvable Machines*, 4:383–403. Kluwer Academic Publishers.
- González, F. A., Galeano, J. C., Rojas, D. A., and Veloza-Suan, A. (2005). Discriminating and visualizing anomalies using negative selection and self-organizing maps. In C. Jacob, M. L. Pilat, and P. J. Bentley (Eds), *GECCO 2005: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation*, volume 1, pages 297–304, Washington DC, USA. ACM Press.
- Hamaker, J. S. and Boggess, L. (2004). Non-euclidean distance measures in AIRS, an artificial immune classification system. In *Proceedings of 2004 Congress on Evolutionary Computation (CEC 2004)*, pages 1067–1073, Portland, OR. IEEE Press.
- Hang, X. and Dai, H. (2004). Constructing detectors in schema complementary space for anomaly detection. In Kalyanmoy Deb et al. (Eds), *LNCS 3102, Proceedings of GECCO*, pages 275–286. Springer.
- Hang, X. and Dai, H. (2005). Applying both positive and negative selection to supervised learning for anomaly detection. In *GECCO 2005: Proceedings of the 2005 conference on Genetic and evolutionary computation*, volume 1, pages 345–352, Washington DC, USA. ACM Press.
- Harmer, P., Williams, G., Gnush, P. D., and Lamont, G. (2002). An artificial immune system architecture for computer security applications. *IEEE Transaction on Evolutionary Computation*, 6(3):252–280. IEEE Press.
- Hart, E. (2005). Not all balls are round: An investigation of alternative recognition-region shapes. In C. Jacob, M. L. Pilat, P. J. Bentley, and J. Timmis (Eds), *ICARIS*, pages 29–42. Springer.
- Hart, E. and Ross, P. (2004). Studies on the implications of shape-space models for idiotypic networks. In G. Nicosia, V. Cutello, P. J. Bentley, and J. Timmis (eds), *Proceedings of Third International Conference on Artificial Immune Systems (ICARIS 2004)*, pages 413–426. Springer.
- Hofmeyr, S. A. (1999). *An Immunological Model of Distributed Detection and its Application to Computer Security*. PhD thesis, University of New Mexico.
- Hofmeyr, S. A. and Forrest, S. (2000). Architecture for an artificial immune system. *Evolutionary Computation*, 7(1):45–68.
- Ji, Z. and Dasgupta, D. (2004). Real-valued negative selection algorithm with variable-sized detectors. In Kalyanmoy Deb et al., (Eds), *LNCS 3102, Proceedings of GECCO*, pages 287–298. Springer.
- Ji, Z. and Dasgupta, D. (2005). Estimating the detector coverage in a negative selection algorithm. In Hans-Georg Beyer et al. (Eds), *GECCO 2005: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation*, volume 1, pages 281–288, Washington DC. ACM Press.
- Ji, Z. and Dasgupta, D. (2006). Applicability issues of the real-valued negative selection algorithms. In Maarten Keijzer et al. (Eds), *GECCO 2006: Proceedings of the 2006 conference on Genetic and evolutionary computation*, volume 1, pages 111–118, Seattle, WA. ACM.

- Kaers, J., Wheeler, R., and Verrelst, H. (2003). The effect of antibody morphology on non-self detection. In J. Timmis, P. Bentley, and E. Hart (Eds), *Proceedings of Second International Conference on Artificial Immune System (ICARIS 2003)*. Springer.
- Kim, J. and Bentley, P. (2002). Immune memory in the dynamic clonal selection algorithm. In J. Timmis, and P.J. Bentley (Eds), *Proceedings of the 1st International Conference on Artificial Immune Systems (ICARIS)*, volume 1, pages 59–67, University of Kent at Canterbury.
- Kim, J. and Bentley, P. J. (2001). An evaluation of negative selection in an artificial immune system for network intrusion detection. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2001)*. Morgan Kaufmann.
- Lee, D.-W. and Sim, K.-B. (2004). Negative selection for DNA sequence classification. In *Proceedings of Joint 2nd International Conference on Soft Computing and Intelligent Systems and 5th International Symposium on Advanced Intelligent Systems (SCIS & ISIS 2004)*, Yokohama, Japan.
- Luo, W., Zhang, Z., and Wang, X. (2006). A heuristic detector generation algorithm for negative selection algorithm with hamming distance partial matching rule. In *Proceedings of ICARIS 2006*, pages 229–243.
- Percus, J. K., Percus, O., and Perelson, A. S. (1993). Predicting the size of the antibody combining region from consideration of efficient self/non-self discrimination. In *Proceedings of the National Academy of Science*, volume 90.
- Shapiro, J. M., Lamont, G. B., and Peterson, G. L. (2005). An evolutionary algorithm to generate hyper-ellipsoid detectors for negative selection. In Hans-Georg Beyer et al. (Eds), *GECCO 2005: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation*, volume 1, pages 337–344, Washington DC. ACM Press.
- Singh, S. (2002). Anomaly detection using negative selection based on the r-contiguous matching rule. In J. Timmis, and P.J. Bentley (Eds), *Proceedings of the 1st International Conference on Artificial Immune Systems (ICARIS)*, volume 1, pages 99–106, University of Kent at Canterbury.
- Stibor, T., Bayarou, K. M., and Eckert, C. (2004). An investigation of r-chunk detector generation on higher alphabets. In Kalyanmoy Deb et al. (Eds), *LNCS 3102, Proceedings of the Conference on Genetic and Evolutionary Computation*, pages 299–307. Springer.
- Stibor, T., Mohr, P., Timmis, J., and Eckert, C. (2005a). Is negative selection appropriate for anomaly detection? In Maarten Keijzer et al. (Eds), *GECCO 2005: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation*, volume 1, pages 321–328, Washington DC. ACM Press.
- Stibor, T., Timmis, J., and Eckert, C. (2005b). A comparative study of real-valued negative selection to statistical anomaly detection techniques. In Christian Jacobet al. (Eds), *ICARIS*, pages 262–275. Springer.
- Stibor, T., Timmis, J., and Eckert, C. (2006a). The link between r-contiguous detectors and k-cnf satisfiability. In *Congress On Evolutionary Computation – CEC*, pages 491–498. IEEE Press. Revised and extended version.
- Stibor, T., Timmis, J., and Eckert, C. (2006b). On permutation masks in hamming negative selection. In H. Bersini, J. Carneiro (Eds), *Proceedings of ICARIS 2006*, pages 122–135. Springer Berlin/ Heidelberg.
- Taylor, D. W. and Corne, D. W. (2003). An investigation of the negative selection algorithm for fault detection in refrigeration system. In J. Timmis, P. Bentley, and E. Hart (Eds), *Proceedings of Second International Conference on Artificial Immune System (ICARIS 2003)*. Springer.
- Wierczon, S. (2000). Discriminative power of the receptors activated by k-contiguous bits rule. *Journal of Computer Science and Technology*, 1(3):1–13.