Natural Computing manuscript No.
(will be inserted by the editor)

Foundations of r-contiguous Matching in Negative Selection for Anomaly Detection

Thomas Stibor

Received: date / Accepted: date

Abstract Negative selection and the associated r-contiguous matching rule is a popular immune-inspired method for anomaly detection problems. In recent years, however, problems such as scalability and high false positive rate have been empirically noticed. In this article, negative selection and the associated r-contiguous matching rule are investigated from a pattern classification perspective. This includes insights in the generalization capability of negative selection and the computational complexity of finding r-contiguous detectors.

Keywords Artificial Immune Systems · Negative Selection · Anomaly Detection · k-CNF Satisfiability

1 Introduction

Theoretical immunologists proposed the r-contiguous matching rule to quantify the binding strength between antibodies and antigens in immune system models [27]. In these models, two strings of the same length have an r-contiguous match, if at least r contiguous characters in both strings are identical. In the field of artificial immune systems, the r-contiguous matching rule is frequently applied as a matching rule for change detection [13] or more generally, for anomaly detection problems [10]. In these problem domains, antibodies (called detectors) and antigens (samples to classify) are abstracted as bit strings and the r-contiguous matching rule is applied as a closeness measure to detect anomalous antigens. To be more precise, the detectors are generated in a censoring process called *negative selection*, such that no detector matches with any normal bit string [17]. The generated detectors are then applied as detection units (similar to antibodies in the immune system) to classify bit strings. A bit string b is classified as anomalous if an r-contiguous match between detector and b occurs, and otherwise as normal.

Thomas Stibor
Technische Universität Darmstadt
Fachbereich Informatik
Hochschulstr. 10, Darmstadt, 64289, Germany
E-mail: stibor@sec.informatik.tu-darmstadt.de

In recent years, attempts were made [13, 42, 41, 2] to generate detectors efficiently, i.e. in polynomial time and with polynomial space occupation with regard to the detector matching length r and number of normal bit strings $|\mathcal{S}|$. All proposed algorithms for generating detectors either have a time or a space complexity which is exponential in the matching length r, i.e. $\mathcal{O}(2^r)$ or in the number of normal bit strings $|\mathcal{S}|$, i.e. $\mathcal{O}(e^{|\mathcal{S}|})$.

Nevertheless the negative selection method was applied on detection problems, such as tool breakage and fault detection [8, 38], novelty detection in time series [9] and (network) intrusion detection [24, 32, 3, 21].

Esponda et al. and Wierzchoń investigated the coverage properties of the r-contiguous matching rule [14, 41, 42]. Empirical studies of the coverage properties are investigated in [22]. However, until now there has been limited available theoretical work on negative selection from the perspective of a *pure pattern classification* problem and *computational complexity* of finding r-contiguous detectors.

In this article, negative selection and the associated r-contiguous matching rule are investigated from a pattern classification perspective. This includes insights in the generalization capability of negative selection and the computational complexity of finding r-contiguous detectors. The article is organized as follows: in section 2 the anomaly detection problem is motivated and in section 3 the immunological principle of negative selection is briefly explained. In sections 3.1 - 3.5 the negative selection algorithm for anomaly detection is presented and generalization capabilities are discussed. Experiments on the generalization capabilities are presented in section 4. Based on the matching probability of two randomly drawn bit strings, a random search approach to generate detectors and the resulting implications are shown in sections 5 - 5.4. The problem equivalence of generating r-contiguous detectors and satisfying Boolean formulas in conjunctive normal form is presented in section 6. In sections 6.2, 6.1 6.2, 6.3 and 7 the problem equivalence is used to explore the computational complexity and the feasibility of generating detectors.

2 Anomaly Detection

Anomaly detection, also referred to as one-class learning or novelty detection is an in-balanced two-class pattern classification problem. That is, training data consists either of examples from a single class C_0 of normal examples, or C_0 *and* a strongly under-represented second class C_1 which contains anomalous examples (see Fig. 1). The test data contains (unseen) samples from both classes. In a probabilistic sense, anomaly detection is equivalent to deciding whether an unknown test sample is produced by the underlying probability distribution that corresponds to the training set of normal examples. This is based on the assumption that the anomalous data is not generated by the source of normal data (see Fig. 2). Popular statistical methods for anomaly detection are for instance non-parametric density estimation techniques [4, 36]. The underlying density is approximated with a Parzen window estimator and a test sample is classified as anomalous if the density of the test sample lies below a predefined threshold. Instead of approximating the full density, one can detect anomalies by assuming that anomalies are not concentrated. This leads to the problem of finding regions where most of the normal data is concentrated [33]. Within the framework of Support Vector Machines, this problem can be formulated in terms of finding in high dimensional feature space the minimum enclosing hypersphere which captures most of

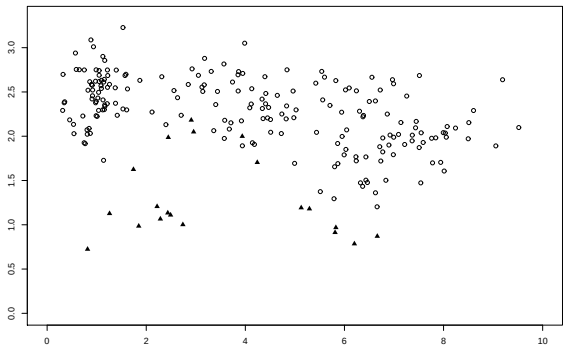


Fig. 1 A "typical" anomaly detection problem with two given classes (C_0 = circles and C_1 = triangles). The number of anomalous examples is strongly under-represented (20 examples with class label C_1) compared to 200 examples of normal data (class label C_0).

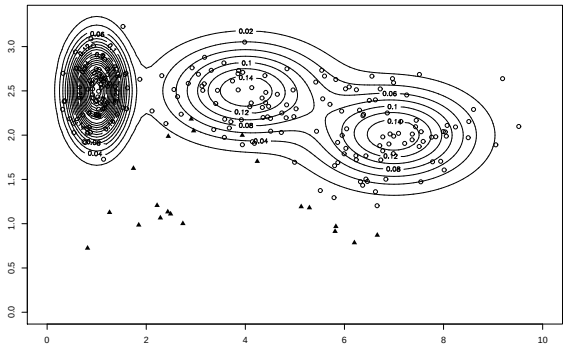


Fig. 2 The underlying probability distribution of class C_0 is depicted as a density plot. One can see that the anomalous data is not generated by the underlying probability distribution of class C_0.

the normal data [37] or, finding a hyperplane which separates the normal data from the origin [30].

In the next section we briefly explain the immunological process of negative selection and show according to [17] how it can be abstracted and formalized to solve an anomaly detection problem.

3 Negative Selection

Negative selection is a process in the immune system to protect the body against developing self-reactive lymphocytes. Lymphocytes carry recognition units (called antibodies) on their surface and are subdivided in two different classes: B and T lymphocytes. The immunological process of negative selection occurs within the thymus on T

lymphocytes only. The thymus forms a highly impermeable barrier to macromolecules called the blood-thymic barrier. The blood-thymic barrier allows thymocytes (immature T lymphocytes) to mature and undergo selection in an environment protected from contact with foreign antigens. During the selection process, antigen presenting cells present self-peptide/major histocompatability complex (MHC) to the T lymphocytes. Those that react strongly (bind with high affinity) with the self-peptide/MHC complexes are eliminated through a controlled cell death called apoptosis. As a result, only those T lymphocytes remain, which can recognize foreign antigens and are not self-reactive. One can say that the negative selection allows the immune system to distinguish between *self* antigens and *non-self* antigens by generating T lymphocytes which can recognize only non-self antigens.

This process has formed the foundation for a large amount of work in the field of artificial immune systems (AIS). In AIS processes and principles of the immune system are abstracted and applied for solving computational problems. In the following sections 3.1 and 3.2 the immunological negative selection process is abstracted and formalized to solve an anomaly detection problem.

3.1 Bit String Matching Rule

A bit string matching rule in the context of AIS is an abstract affinity measure between antibodies and antigens.

Let $\mathcal{U}$ be a universe which contains all 2^l distinct bit strings of length l.

Definition 1 A bit string $b \in \mathcal{U}$ with $b = b_1 b_2 \dots b_l$ and detector $d \in \mathcal{U}$ with $d = d_1 d_2 \dots d_l$, match with r-contiguous rule, if a position p exists where $b_i = d_i$ for $i = p, \dots, p + r - 1$ and $p \leq l - r + 1$.

Loosely speaking, two bit strings, with the same length, match if at least r contiguous bits are identical. The r-contiguous matching rule is used primarily in negative selection [8, 9, 38, 24, 32, 2]. Other affinity measures used in the field of AIS are Hamming, Rogers-Tanimoto and r-chunk matching rules [22, 6].

In the remaining sections the expression "detectors" will refer to r-contiguous detectors. Sets are denoted in calligraphic letters, e.g. $\mathcal{S}$ and $|\mathcal{S}|$ denotes the cardinality. Throughout the paper, if not otherwise stated, we will assume that $\mathcal{S}$ contains pairwise distinct bit strings randomly drawn from $\mathcal{U}$.

3.2 Negative Selection Algorithm

Given $\mathcal{U}$ and $\mathcal{S}$, in negative selection one has to find[1] detectors such that no detector matches (see Def. 1) with any bit string from $\mathcal{S}$. Detectors which satisfy this property match with — not necessarily all — bit strings from the complementary space $\mathcal{U} \setminus \mathcal{S}$. After a detector set $\mathcal{D}$ is generated, (unseen) bit strings $\delta \subseteq \mathcal{U}$ are matched against the bit strings of $\mathcal{D}$ and classified as anomalous if an r-contiguous match occurs, otherwise as normal bit strings (see Alg. 1 and Fig. 3)

[1] "To find" or "to generate" detectors means the same in this article.

Algorithm 1: Negative Selection Algorithm.

input : $\mathcal{S} \subseteq \mathcal{U} \equiv$ normal class training examples, $r \in \mathbb{N} \equiv$ matching length
begin
  Generate a set $\mathcal{D}$ of detectors, such that each fails to match any element in $\mathcal{S}$.
  Monitor data $\delta \subseteq \mathcal{U}$ by continually matching the detectors in $\mathcal{D}$ against δ. If any detector matches with δ, classify δ as an anomaly, otherwise as normal.
end

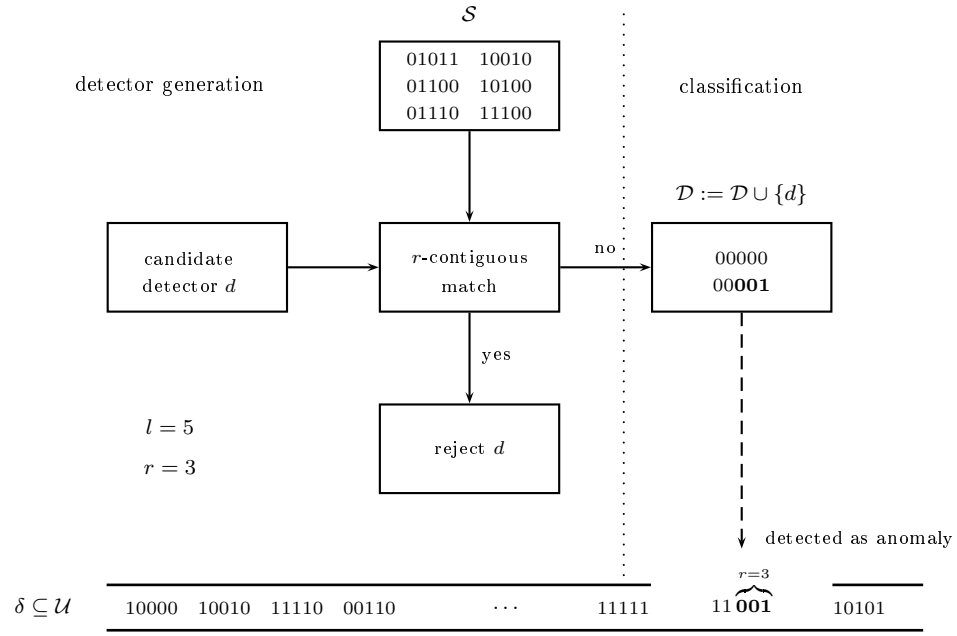


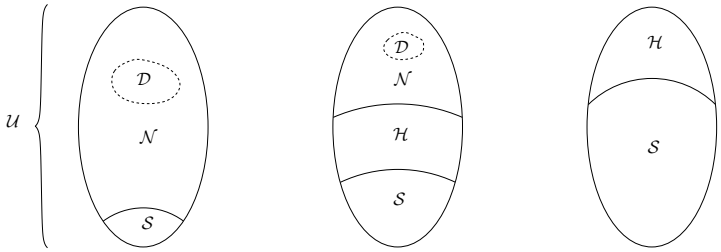
Fig. 3 Principle of negative selection. Detectors are generated in a censoring process called negative selection such that no detector matches with any bit strings of $\mathcal{S}$. Bit strings $\delta \subseteq \mathcal{U}$ are then classified with the generated detectors as anomalous if an r-contiguous match occurs, otherwise as normal bit strings.

3.3 Partition of Universe $\mathcal{U}$ and Undetectable Bit Strings

Through the application of the r-contiguous matching rule in negative selection, the universe $\mathcal{U}$ is partitioned in distinct subsets. Assume that $|\mathcal{S}| < \nu$, that is, $\mathcal{S}$ contains less than some threshold ν of bit strings and let $\mathcal{D}$ be the set of all detectors that can be generated. In this case the following coherence holds:

$$\mathcal{U} = \mathcal{N} \cup \mathcal{S}, \quad \mathcal{S} \cap \mathcal{N} = \emptyset \quad \text{and} \quad \mathcal{D} \subseteq \mathcal{N}.$$

The detector set $\mathcal{D}$ is a subset of $\mathcal{N}$, where $\mathcal{N}$ is the set of detectable, i.e. covered bit strings (see Fig. 4(a)). However, if $|\mathcal{S}| \geq \nu$ then an additional set is induced, namely the set $\mathcal{H}$ of undetectable bit strings called holes [13]. $\mathcal{H}$ contains bit strings which are *not* members of $\mathcal{S}$ and $\mathcal{N}$ (see Fig. 4(b)) and hence cannot be detected by any detector.



(a) The universe $\mathcal{U}$ is partitioned in set $\mathcal{S}$ and $\mathcal{N}$ only. The detectors of $\mathcal{D} \subseteq \mathcal{N}$ cover *all* bit strings of $\mathcal{N}$.

(b) $\mathcal{S}$ contains more then ν bit strings and this induces the set $\mathcal{H}$ of undetectable bit strings.

(c) $\mathcal{S}$ contains such a large number of distinct bit strings and therefore $\mathcal{U}$ is only partitioned in $\mathcal{S}$ and $\mathcal{H}$, i.e. no detectors exist and hence, all bit strings of $\mathcal{U}$ are undetectable.

Fig. 4 Coherences of cardinalities of sets $\mathcal{S}$,$\mathcal{N}$,$\mathcal{H}$ and $\mathcal{D}$.

More specifically, the following coherence holds:

$$\mathcal{U} = \mathcal{N} \cup \mathcal{S} \cup \mathcal{H} \quad \text{where}$$
$$\mathcal{N} \cap \mathcal{S} = \emptyset, \quad \mathcal{N} \cap \mathcal{H} = \emptyset, \quad \mathcal{H} \cap \mathcal{S} = \emptyset \quad \text{and}$$
$$\mathcal{D} \subseteq \mathcal{N}.$$

If $|\mathcal{S}| \gg \nu$, then the universe $\mathcal{U}$ will consist only of the sets $\mathcal{S}$ and $\mathcal{H}$ (see Fig. 4(c)), that is no detectors can be generated.

Example 1 Let $l = 4$, $r = 2$ and $\mathcal{S} = \{s_1, s_2, s_3\}$, where $s_1 = \{0110\}$, $s_2 = \{1010\}$ and $s_3 = \{1100\}$. One can easily verify that only *one* detector can be generated, namely $\{0001\}$. That implies that all bit strings of set[2] $\mathcal{N} = \{00{**}, {*}00{*}, {**}01\}$ are detectable. Conversely, all bit strings from $\mathcal{U} \backslash \mathcal{N} = \mathcal{S} \cup \mathcal{H} = \{s_1, s_2, s_3, 0100, 0111, 1011, 1110, 1111\}$ are not detectable.

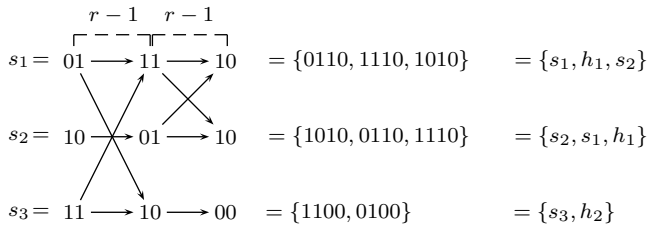
By using the same parameters l, r and adding one additional normal bit string $s_4 = \{0011\}$ to $\mathcal{S}$, no detectors can be generated, that is $\mathcal{D} = \emptyset$ and hence $\mathcal{N} = \emptyset$.

3.4 Constructing Holes with the Crossover Closure

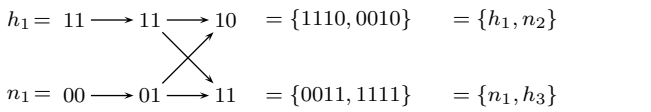
Holes can be constructed by the crossover closure method proposed in [15]. The idea behind the crossover closure is presented in figure 5. Each bit string $s \in \mathcal{S}$ is subdivided in $l - r + 1$ substrings[3] $s[1, \ldots, r], s[2, \ldots, r + 1], \ldots, s[l - r + 1, \ldots, l]$ and connected with a direct edge, if the last $r - 1$ bits of $s[i, \ldots, r + i - 1]$ are matched with the first bits of $s[i + 1, \ldots, r + i]$, for $i = 1, \ldots, l - r$ and all $s \in \mathcal{S}$. Substrings which are connected with a direct edge are merged over $r - 1$ equal bits to one bit string of length l. By applying the construction method on bit strings of $\mathcal{S}$, only holes can be constructed which are "crossed" combinations of bit strings of $\mathcal{S}$. To construct *all*

[2] The symbol * represents either a 1 or 0.

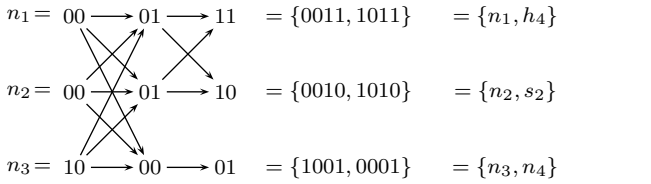
[3] $s[1, \ldots, l]$ denotes characters of s at positions $1 \ldots l$.



(a) Holes ($h_1 = 1110$, $h_2 = 0100$) can be constructed by $s_1 = 0110, s_2 = 1010$ and $s_3 = 1100$.



(b) An additional hole h_3 can be constructed by the already found hole $h_1 = 1110$ and $n_1 = 0011$.



(c) Hole $h_4 = 1011$ can be constructed by $n_1 = 0011$, $n_2 = 0010$ and $n_3 = 1001$.

Fig. 5 Holes constructed by means of the crossover closer method for bit strings from $\mathcal{N}, \mathcal{S}$ and $\mathcal{H}$.

existing holes, one has to include in the crossover closure method also bit strings of $\mathcal{N}$ and already constructed holes of $\mathcal{H}$. To clarify this fact and to visualize the crossover closure method, consider again example 1 with parameters $l = 4$, $r = 2$ and $\mathcal{S} = \{s_1, s_2, s_3\}$. In figure 5(a) one can see, that holes $h_1 = 1110$ and $h_2 = 0100$ can be constructed by $s_1 = 0110, s_2 = 1010$ and $s_3 = 1100$. Moreover as illustrated in figure 5(b), the additional hole $h_3 = 1111$ can be constructed by bit string $n_1 = 0011$ and hole $h_1 = 1110$. Hole $h_4 = 1011$ can be constructed by n_1, n_2 and n_3 (see Fig. 5(c)). The remaining hole ($h_5 = 0111$) can be constructed by applying the crossover closure method on bit strings s_1 and h_3. This example illustrates that holes are *not only* induced by bit strings of $\mathcal{S}$, but also by bit strings of $\mathcal{U} \setminus \mathcal{S}$.

3.5 Holes as Generalization

Holes are undetectable bit strings and hence have to represent unseen bit strings of $\mathcal{S}$ to generalize beyond the training set. The number of holes is determined by $|\mathcal{S}|$ and matching length r. A detector set which generalizes well, ensures that seen and unseen bit strings of $\mathcal{S}$ are *not* recognized by any detector, whereas all other bit strings are recognized by detectors and classified as anomalous. A detector set which covers all bit strings of $\mathcal{N}$ and all unseen bit strings of $\mathcal{S}$ *overfits*, because no holes (no generalization) exists. In contrast, the opposite result can occur, that is, a large number of anomalous bit strings are members of $\mathcal{H}$ and hence the detector set consequently *underfits*. To summarize, in order to obtain good generalization results, it is crucial to find proper parameter combinations of $|\mathcal{S}|$ and r, such that the generated detector set generalizes well. This also includes the topological regions of holes, that is, holes must occur in regions where most normal bit strings are concentrated.

4 Experiments on Covered Regions and Holes

In order to analyze the generalization capabilities with regard to parameters $|\mathcal{S}|$ and r, two artificially generated data sets are created. Both data sets consists of only normal examples which where generated by an underlying mixture of Gaussian distributions with different mean vectors and covariance matrices. The density plots of both probability distributions and generated normal examples are depicted in figure 6. As the negative selection operates on bit strings, and examples from data sets 1 and 2 are two-dimensional real numbers, both data sets are min-max normalization to $[0,1]^2$ and discretized to binary strings of length $l = 16$

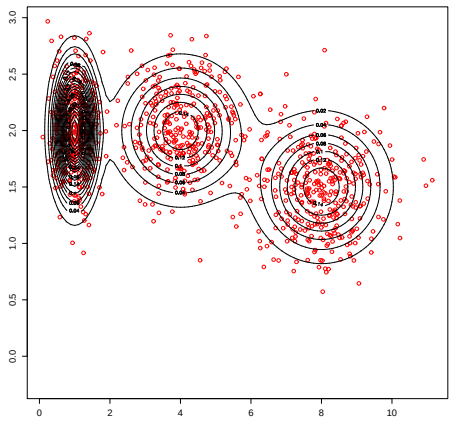
$$\underbrace{b_1, b_2, \ldots, b_8}_{b_x}, \underbrace{b_9, b_{10}, \ldots, b_{16}}_{b_y},$$

where the first 8 bits encode the integer x-value $i_x := \lceil 255 \cdot x + 0.5 \rceil$ and the last 8 bits the integer y-value $i_y := \lceil 255 \cdot y + 0.5 \rceil$, i.e.

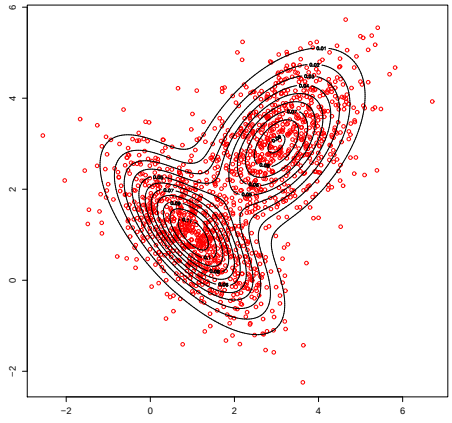
$$[0,1]^2 \rightarrow (i_x, i_y) \in (1, \ldots, 256) \times (1, \ldots, 256) \rightarrow (b_x, b_y) \in \{0,1\}^8 \times \{0,1\}^8.$$

This mapping is proposed in [22], and also utilized in [34]. It satisfies a straightforward visualization of real-valued encoded points in negative selection.

In the appendix (see Figs. 14, 15, 16, 17) the experimental results on the covered regions with detectors and holes are visualized for different parameter combinations of $|\mathcal{S}|$ and r. One can see that the number of holes is determined by the cardinality of $\mathcal{S}$ and the matching length r. Given approximately 250 normal bit strings, no detectors can be generated for matching lengths $r = \{5, 6\}$, whereas for approximately 5000 normal bit strings no detectors can be generated for $r = \{5, 6, 7\}$ (resp. $r = \{5, 6, 7, 8\}$ for data set 2). By increasing stepwise the value of r to the maximum value of $r = 16$, one can observe that the cardinality of $\mathcal{N}$, that is, the set which is covered with detectors increases and in contrast $|\mathcal{H}|$ decreases. Moreover, one can also observe that for approximately 250 normal bit strings the cardinality variation of $\mathcal{N}$ and $\mathcal{H}$ for stepwise increasing the value of r, rapidly occurs, that is, for $r \leq 6$ no regions are covered by detectors, whereas for $r \geq 9$ almost all regions are covered. This sharp



(a)



(b)

Fig. 6 Normal examples of data set 1 (left figure) and 2 (right figure) are sampled from a mixture of Gaussian distributions which are depicted as density plots. Normal data is concentrated within the high density regions and hence holes must occur within these regions.

phase transition shift of the cardinalities is more closely investigated in section 5.2 and 6.2.

Furthermore, one can additionally observe that holes never occur in dense normal regions only, or to say it the other way around, the generated detectors are not capable of covering only anomalous regions and hence the detector set does not generalize well. This observation is obviously biased by the r-contiguous matching rule. To be more precise, the inductive bias of negative selection and the associated r-contiguous matching rule is the assumption that bit string b belongs to the normal class, if b and the given normal bit strings of $\mathcal{S}$ have at least r bits in common. Comparing a consecutive number of bits as a closeness measure seems not to be an appropriate approach, because

the semantic representation of the underlying data can not be properly captured, or in other words, the inductive bias cannot be learned. The r-contiguous matching rule and three additional rules (Hamming, Rogers-Tanimoto and r-chunk) are tested in a similar experiment [22] and reveals comparable results, even if the representation is encoded in Gray codes. The crucial fact of having a proper inductive bias is well known in the machine learning community [25], however in the context of pattern classification and negative selection, it was only discussed in [19].

5 Generating Detectors Randomly

A straightforward approach to generate detectors is to randomly sample a bit string d from $\mathcal{U}$ and to match d against all bit strings in $\mathcal{S}$. When no r-contiguous match occurs, d is added to the detector set $\mathcal{D}$ [17]. This random sampling is repeated until a certain number of detectors is found (see algorithm 2). It is obvious that this straightforward random search is not an efficient search technique.

However, a thorough probabilistic analysis of algorithm 2 reveals valuable insights, whether detectors can or can not be generated and how $\mathcal{U}$ is partitioned with respect to parameters $|\mathcal{S}|, l$ and r.

Algorithm 2: Random search for detectors in negative selection

input : $l, r, t \in \mathbb{N}$ where $1 \leq r \leq l$ and $\mathcal{S} \subset \mathcal{U}$
output: Set $\mathcal{D} \subset \mathcal{U}$ of r-contiguous detectors

```
1 begin
2     D := ∅
3     while |D| < t do
4         Sample randomly a bit string d ∈ U
5         if d does not match with any bit string of S then
6             D := D ∪ {d}
7 end
```

5.1 Probability of Matching in Random Detector Generation

The probability that two randomly drawn bit strings from $\mathcal{U}$ are *not* matching with the r-contiguous rule can be determined with approaches from probability theory, namely recurrent events and renewal theory [16]. In Feller's textbook on probability theory an equivalent[4] problem is formulated as follows:

> "*A sequence of n letters S and F contains as many S-runs of length r as there are non-overlapping uninterrupted blocks containing exactly r letters S each*".

[4] The Link between recurrent events, renewal theory and the r-contiguous matching probability was discovered originally in [27] and rediscovered in [28]. Percus et al. presented in [27] the probability approximation (2) which is only valid for $r \geq l/2$. However, they also cited Uspensky's textbook (see pp. 77 in [39]), where the approximation of the r-contiguous matching probability for $1 \leq r \leq l$ is presented.

Given a Bernoulli trial with outcomes S (success) and F (failure), the probability of no success running of length r in l trials is according to Feller

$$P = \frac{1 - px}{(r + 1 - rx)q} \cdot \frac{1}{x^{l+1}} \tag{1}$$

where

$$p = q = \frac{1}{2} \quad \text{and} \quad x = 1 + qp^r + (r+1)(qp^r)^2 + \dots$$

A simpler approximation — however only valid for $r \geq l/2$ (see [41,35]) — is provided in [27]:

$$\widehat{P} = 1 - 2^{-r} \left[(l - r)/2 + 1\right]. \tag{2}$$

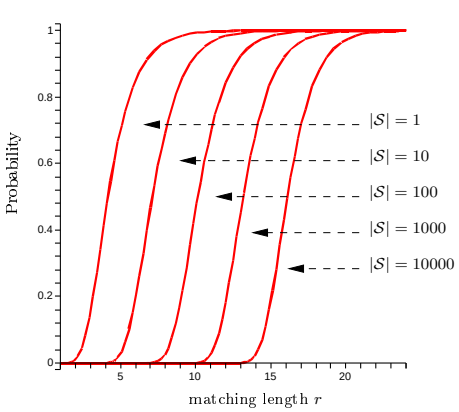
From (1) one can straightforwardly conclude that the probability of finding t detectors when given l, r and $|\mathcal{S}|$ results in:

$$\textbf{Prob}[\text{find } t \text{ detectors}] = t^{-1} \cdot P^{|\mathcal{S}|}. \tag{3}$$

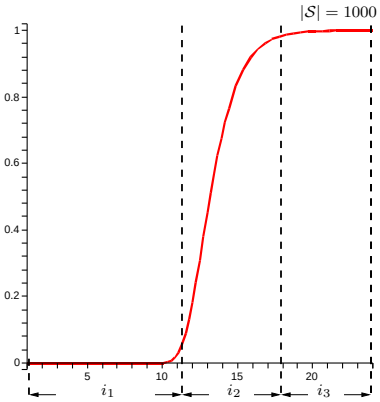
Moreover, from (3) one can conclude how often on average step 4 in algorithm (1) is executed when given t, or in other words how many bit strings one has to sample before finding $\widehat{t}$ detectors.

$$\widehat{t} = \frac{1}{t^{-1} \cdot P^{|\mathcal{S}|}}. \tag{4}$$

Result (4) is equivalent to an earlier result provided in [17], when P is replaced by $\widehat{P}$.



(a) Matching probability for finding a detector randomly for $l := 24$, $r := \{1, 2, \dots, 24\}$ and $|\mathcal{S}| := \{1, 10, 100, 1000, 10000\}$.



(b) If r lies within interval i_1, then with high probability no detectors will be found, whereas if r lies within interval i_3, then with high probability, detectors will be found. There also exists an interval i_2 where the probability rapidly changes from 0 to 1.

Fig. 7 Coherence between the probability of finding a detector randomly and the parameters l, r and $|\mathcal{S}|$. There exists a sharp transition boundary where the probability rapidly changes from 0 to 1.

5.2 Probability Transition in r-contiguous Matching

Knowing the probability P enables us to investigate the combinations of parameters $|\mathcal{S}|, l$ and r where, with high probability detectors can be generated or with high probability can not be generated. The graphs in figure 7 show the probability for finding a detector for fixed l and variable r and $|\mathcal{S}|$ according to term (3). One can see, that the larger the cardinality of $\mathcal{S}$, the larger the interval for r where the resulting probability is nearly 0 to find a detector. On the other hand, the smaller the cardinality of $\mathcal{S}$, the larger the interval for r where the resulting probability is nearly 1 to find a detector. In figure 7(b) the graph for $|\mathcal{S}| = 1000$ is highlighted. One can see in detail that three different intervals (i_1, i_2, i_3) exist. One can either find with high probability a detector if r falls in interval i_1, or find with high probability no detector if r falls in interval i_3. Moreover, there exists a third interval i_2 where the probability rapidly changes from 0 to 1. For the sake of conformity with the subsequent sections, we denote the interval i_2 as *phase transition* region. We will later see that finding detectors in this region, which is characterized by certain combinations of parameters $|\mathcal{S}|, l, r$ is hardest from the perspective of computational complexity.

To summarize this section, if parameters $|\mathcal{S}|, l$ and r are chosen such that term (3) results in a value very close to 0, then in the worst case no detectors can be generated, never mind which algorithms, i.e. search techniques are applied to generate detectors, because there exist no detectors. On the other hand, if term (3) is close to 1, then a large number of detectors exist.

5.3 Coherence of Matching Length r, Self Set $\mathcal{S}$ and Random Detector Search

In the AIS community there seems to exist some confusion regarding the time complexity of algorithm (1). [17] argued that generating detectors when applying the random search approach can be performed linearly in $|\mathcal{S}|$. Their argument is based on the observation that $\widehat{t}$ in (4) is minimized by choosing $1-\widehat{P} \approx 1/|\mathcal{S}|$. In other words, the number of bit strings one has to sample before finding t detectors is linear proportionally to $|\mathcal{S}|$, when using algorithm (1). This observation implies that the matching length r purely depends on the cardinality of $\mathcal{S}$ when l is fixed. To be more precise, suppose $r \geq l/2$, then

$$2^{-r}\left[(l-r)/2+1\right] \approx |\mathcal{S}|^{-1} \iff \frac{8 \cdot 2^l}{|\mathcal{S}|} \approx (l-r+2) \cdot 2^{l-r+2} \tag{5}$$

$$\iff \frac{8\ln(2)2^l}{|\mathcal{S}|} \approx (l-r+2)\ln(2)\,\mathrm{e}^{(l-r+2)\ln(2)} \tag{6}$$

$$\iff r \approx l+2-\frac{W(8\ln(2)2^l/|\mathcal{S}|)}{\ln(2)} \tag{7}$$

where $W(x)$ is the Lambert W-function which can be expressed as the series expansion

$$W(x) = \sum_{k=1}^{\infty} \frac{(-1)^{k-1}k^{k-2}}{(k-1)!}x^k \tag{8}$$

and provides a solution to the problem $Y = X\mathrm{e}^X \iff X = W(Y)$. Practically speaking, once $|\mathcal{S}|$ and l are fixed, the matching length r is chosen according to (7) and this

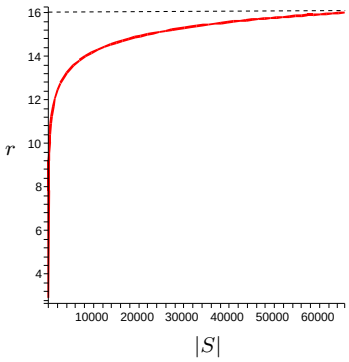


Fig. 8 Coherence of matching length r and growing cardinality of $\mathcal{S}$ in Eq. 7 for $l = 16$.

consequently implies that r grows exponentially in $|\mathcal{S}|$ and "quickly" approaches to l (see Fig. 8), that is

$$\lim_{|\mathcal{S}|\to 2^l} l + 2 - \frac{W(8\ln(2)2^l/|\mathcal{S}|)}{\ln(2)} = l + 2 - \underbrace{\frac{W(8\ln(2))}{\ln(2)}}_{2} = l. \tag{9}$$

In terms of the inductive bias of the r-contiguous matching rule, the dependence between r and $|\mathcal{S}|$ in (7) is problematic because the value of r is inextricably linked to the underlying data being analyzed. To be more precise, the value of r must capture the underlying features of the data — Freitas and Timmis termed this, the positional bias [19].

According to assumption $1 - \widehat{P} \approx 1/|\mathcal{S}|$ and term (7), the value of r is however determined independently of the underlying data — the value of r depends only on the cardinality of $\mathcal{S}$. Let us assume that $\mathcal{S}$ contains 1000 bit strings sampled from some distribution P, and the value of r is appropriately determined, that is, it captures semantically the features of the underlying data. Let us now assume that $\mathcal{S}$ contains 2000 bit strings sampled from P, it is clear that the value of r has to be equal to the value of r which is determined previously for 1000 bit strings. According to assumption $1 - \widehat{P} \approx 1/|\mathcal{S}|$ and term (7) however, the value of r depends only on the cardinality of $\mathcal{S}$ rather than the semantics of the underlying data.

5.4 Average Number of Detectors and Holes

By applying the results from the previous section 5.1, one can approximate the average number of detectors that can be generated and the number of resulting holes.

Knowing this coherence between term (3) and the universe composition, the average number of detectors that can be generated results in

$$\mathbf{E}[|\mathcal{D}|] = 2^l \cdot P^{|\mathcal{S}|}. \tag{10}$$

As the universe is composed of $\mathcal{U} = \mathcal{S} \cup \mathcal{N} \cup \mathcal{H}$, the number of holes results in

$$|\mathcal{H}| = |\mathcal{U}| - |\mathcal{N}| - |\mathcal{S}| \tag{11}$$

where

$$\mathbf{E}[|\mathcal{N}|] = 2^l \quad - \quad \underbrace{2^l \cdot P^{\mathbf{E}[|\mathcal{D}|]}}_{\substack{\text{Number of bit strings} \\ \text{not detected by } \mathbf{E}[|\mathcal{D}|] \\ \text{detectors}}} \tag{12}$$

and hence the average number of holes results in

$$\mathbf{E}[|\mathcal{H}|] = 2^l \cdot P^{\mathbf{E}[|\mathcal{D}|]} - |\mathcal{S}|. \tag{13}$$

In figure 9, the term (10) and (13) is plotted for $l = 12, r = 7$ and $|\mathcal{S}| := \{1, 2, \ldots, 2^l\}$. Additionally, for each cardinality value of $\mathcal{S}$ (randomly drawn from $\mathcal{U}$) the resulting number of detectors that can be generated and resulting holes (black and gray circles) is empirically determined and depicted. One can see that term (10) and (13) are reasonable estimations of the number of detectors that can be generated as well as the number of resulting holes. Furthermore, one can see the exponential decrease of the number of detectors and as a countermove the increase of holes for $|\mathcal{S}| := \{1, 2, \ldots, 2^l\}$. If the maximum number of possible holes is reached, then $|\mathcal{H}|$ decreases linearly to the value of 0 because the relation $|\mathcal{U}| = |\mathcal{S}| + |\mathcal{N}| + |\mathcal{H}|$ must hold.

6 The Link between r-contiguous Detectors and k-CNF Satisfiability

In this section we outline the Boolean satisfiability problem and subsequently show how detectors are related to that problem.

The Boolean satisfiability problem (short SAT) is a decision problem and can be formulated in terms of the language SAT [7]. An instance of SAT is a Boolean formula ϕ composed of $\wedge$ (AND), $\vee$ (OR), $\bar{\cdot}$ (NOT), $\rightarrow$ (implications), $\leftrightarrow$ (if and only if), variables $x_1, x_2, \ldots$, and parentheses. In SAT problems, one has to decide if there is some assignment of *true* and *false* values to the variables that will make the Boolean formula ϕ true. In the following sections, we will focus on Boolean formulas in conjunctive normal form.

A Boolean formula is in conjunctive normal form (CNF), if it is expressed as an AND-combination of clauses and each clause is expressed as an OR-combination of one or more literals. A literal is an occurrence of a Boolean variable x or its negation $\overline{x}$.

Example 2

$$\underbrace{(\overbrace{x_1}^{\text{literal}} \vee \overline{x}_1 \vee \overline{x}_2)}_{\text{clause}} \wedge (x_3 \vee x_2 \vee x_4) \wedge (\overline{x}_1 \vee \overline{x}_3 \vee \overline{x}_4)$$

A Boolean formula is in k-CNF, if each clause has exactly k distinct literals. Example (2) shows a 3-CNF Boolean formula. A k-CNF Boolean formula is *satisfiable* if there exists a set of values ($0 \equiv$ false and $1 \equiv$ true) for the literals that cause it to evaluate to 1, i.e. the logical value *true*. A possible assignment set of Boolean values that evaluate in example (2) to true is, $x_1 = 1, x_2 = 1, x_3 = 0, x_4 = 0$ (or expressed

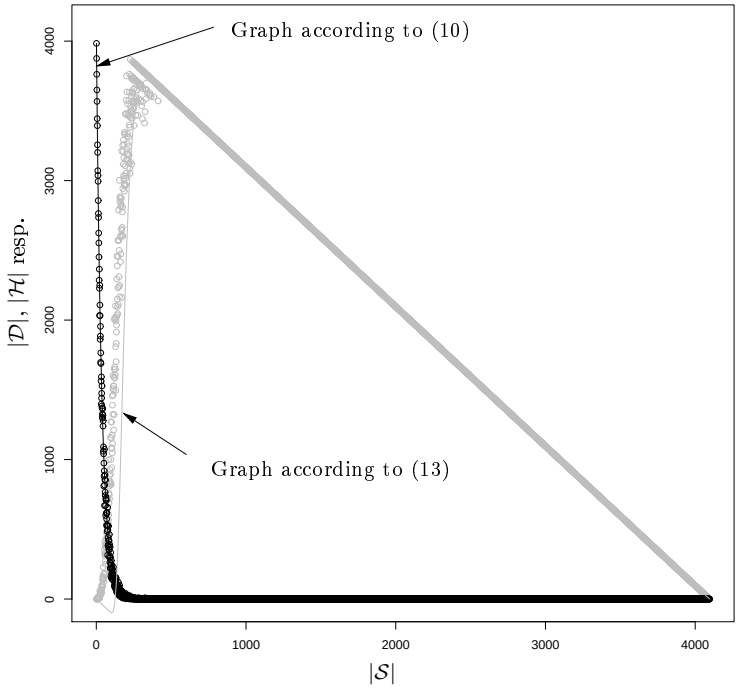


Fig. 9 Coherence between the number of detectors that can be generated and the resulting number of holes for $l = 12, r = 7$ and $|\mathcal{S}| := \{1, 2, \ldots, 2^l\}$ (randomly drawn from $\mathcal{U}$). The black and gray circles are empirically determined values of $|\mathcal{D}|$ and $|\mathcal{H}|$, the black and gray colored graph denotes the analytically determined values according to (10) and (13).

as a bit-string 1100). In k-CNF-SAT, we are asked whether a given Boolean formula in k-CNF is satisfiable. It is known that for $k > 2$, k-CNF-SAT is $\mathcal{NP}$-complete [29], i.e. this problem is *verifiable* in polynomial time, but no-one has yet discovered an algorithm for solving *all* k-CNF-SAT instances in polynomial time.

We will now consider a special subset of Boolean formulas in k-CNF which are defined as follows:

Definition 2 A k-CNF Boolean formula ϕ_{rcb} is in l-k-CNF, when ϕ_{rcb} has $(l-k+1)$ clauses $C_1, C_2, \ldots, C_{l-k+1}$ for $1 \leq k \leq l$ and $k-1$ equal literals in C_i, C_{i+1} for $i = 1, 2, \ldots, l-k$

$$\begin{aligned} C_1 &= (x_1 \vee x_2 \vee \ldots \vee x_k) \\ C_2 &= (x_2 \vee x_3 \vee \ldots \vee x_{k+1}) \\ &\vdots \\ C_{l-k+1} &= (x_{l-k+1} \vee x_{l-k+2} \vee \ldots \vee x_l). \end{aligned}$$

Recall detectors are bit strings of $\mathcal{U}$ which do not match with any bit strings of length l from $\mathcal{S}$. We subsequently show a transformation of bit strings of $\mathcal{S}$ into l-k-CNF

Boolean formulas.

Let $b \in \{0,1\}$ and $\mathfrak{L}(b)$ a mapping defined as:

$$\mathfrak{L}(b) \rightarrow \begin{cases} x & \text{if} \quad b = 0 \\ \overline{x} & \text{otherwise} \end{cases}$$

where $x, \overline{x}$ are literals.

Let $k, l \in \mathbb{N}$, where $k \leq l$ and $s \in \mathcal{U}$, where $s[i]$ denotes the bit at position i of bit-string s, and $\mathfrak{C}(s,k)$ a l-k-CNF mapping defined as:

$$\begin{aligned} \mathfrak{C}(s,k) \rightarrow & (\mathfrak{L}(s[1]) \ \vee \ \mathfrak{L}(s[2]) \ \vee \ \ldots \ \vee \ \mathfrak{L}(s[k])) \ \wedge \\ & (\mathfrak{L}(s[2]) \ \vee \ \mathfrak{L}(s[3]) \ \vee \ \ldots \ \vee \ \mathfrak{L}(s[k+1])) \ \wedge \\ & \qquad \vdots \\ & (\mathfrak{L}(s[l-k+1]) \ \vee \ \ldots \ \vee \ \mathfrak{L}(s[l]))\,. \end{aligned}$$

For the sake of clarity we denote a Boolean formula in l-k-CNF which is obtained by $\mathfrak{C}(s,k)$ for $s \in \mathcal{S}$ as ϕ_{rcb}. Moreover we denote a Boolean formula $\bigwedge_{i=1}^{|\mathcal{S}|} \phi_{rcb}^{i}$ which is obtained by $\mathfrak{C}(s_1,k) \wedge \mathfrak{C}(s_2,k) \wedge \ldots \wedge \mathfrak{C}(s_{|S|},k)$ for $|\mathcal{S}| \geq 1$ and all $s_i \in \mathcal{S}$, $i = 1, \ldots, |\mathcal{S}|$ as $\widehat{\phi}_{rcb}$. If $|\mathcal{S}| = 1$, then $\phi_{rcb} \equiv \widehat{\phi}_{rcb}$.

Proposition 1 *Given $\mathcal{U}$, $\mathcal{S}$ and the set $\mathcal{D}$ which contains all detectors that can be generated. The Boolean formula $\widehat{\phi}_{rcb}$ which is obtained by $\mathfrak{C}(s,r)$ for all $s \in \mathcal{S}$ is satisfiable only with the assignment set $\mathcal{D}$.*

Proof Transforming $s_1 \in S$ with $\mathfrak{C}(s_1,k)$ in a l-k-CNF, where $k := r$, results due to $\mathfrak{L}(\cdot)$ in a Boolean formula which is only satisfiable with bit strings from $\mathcal{U} \setminus \mathcal{F}_1$, where the symbol $*$ represents either a 1 or 0 and

$$\begin{aligned} \mathcal{F}_1 = \{ & s_1[1,\ldots,r] \underbrace{*\,*\ldots*}_{l-r}, \\ & * \, s_1[2,\ldots,r+1] \underbrace{*\,*\ldots*}_{l-r-1}, \\ & \qquad \vdots \\ & \underbrace{*\,*\ldots*}_{l-r} \, s_1[l-r+1,\ldots,l]\}. \end{aligned}$$

Transforming the remaining $s_i = s_2, s_3, \ldots, s_{|S|}$ with $\mathfrak{C}(s_i,k)$ and constructing $\widehat{\phi}_{rcb} = \phi_{rcb}^{1} \ \wedge \ \phi_{rcb}^{2} \ \wedge \ \ldots \ \wedge \ \phi_{rcb}^{|\mathcal{S}|}$ results in a Boolean formula which is only satisfiable with bit strings from $\mathcal{U} \setminus (\mathcal{F}_1 \cup \mathcal{F}_2 \cup \ldots \cup \mathcal{F}_{|\mathcal{S}|})$. Each detector of $\mathcal{D}$ has *no* matching bits at $s_i[1,\ldots,r], s_i[2,\ldots,r+1], \ldots, s_i[l-r+1,\ldots,l]$ for $i = 1,2,\ldots,|\mathcal{S}|$. Hence, $\widehat{\phi}_{rcb}$ is only satisfiable with assignment set $\mathcal{U} \setminus (\mathcal{F}_1 \cup \mathcal{F}_2 \cup \ldots \cup \mathcal{F}_{|\mathcal{S}|}) = \mathcal{D}$.

Example 3 Let $l = 5,\ r = 3$ and $\mathcal{S} = \{s_1, s_2, s_3, s_4, s_5, s_6\}$ with the following bit strings:

$$s_1 = \{01011\},\ s_2 = \{01100\},\ s_3 = \{01110\},$$
$$s_4 = \{10010\},\ s_5 = \{10100\},\ s_6 = \{11100\}.$$

Generating all possible detectors, one obtains the detector set $\mathcal{D} = \{d_1, d_2, d_3, d_4, d_5\}$:

$$d_1 = \{00000\},\ d_2 = \{00001\},\ d_3 = \{11000\},$$
$$d_4 = \{11001\},\ d_5 = \{00111\}.$$

Transforming all $s \in \mathcal{S}$ with $\mathfrak{C}(s, r)$, one obtains:

$$\begin{aligned}
\phi^1_{rcb} &= (x_1 \vee \overline{x}_2 \vee x_3) \wedge (\overline{x}_2 \vee x_3 \vee \overline{x}_4) \wedge \\
&\quad (x_3 \vee \overline{x}_4 \vee \overline{x}_5) \\
\phi^2_{rcb} &= (x_1 \vee \overline{x}_2 \vee \overline{x}_3) \wedge (\overline{x}_2 \vee \overline{x}_3 \vee x_4) \wedge \\
&\quad (\overline{x}_3 \vee x_4 \vee x_5) \\
\phi^3_{rcb} &= (x_1 \vee \overline{x}_2 \vee \overline{x}_3) \wedge (\overline{x}_2 \vee \overline{x}_3 \vee \overline{x}_4) \wedge \\
&\quad (\overline{x}_3 \vee \overline{x}_4 \vee x_5) \\
\phi^4_{rcb} &= (\overline{x}_1 \vee x_2 \vee x_3) \wedge (x_2 \vee x_3 \vee \overline{x}_4) \wedge \\
&\quad (x_3 \vee \overline{x}_4 \vee x_5) \\
\phi^5_{rcb} &= (\overline{x}_1 \vee x_2 \vee \overline{x}_3) \wedge (x_2 \vee \overline{x}_3 \vee x_4) \wedge \\
&\quad (\overline{x}_3 \vee x_4 \vee x_5) \\
\phi^6_{rcb} &= (\overline{x}_1 \vee \overline{x}_2 \vee \overline{x}_3) \wedge (\overline{x}_2 \vee \overline{x}_3 \vee x_4) \wedge \\
&\quad (\overline{x}_3 \vee x_4 \vee x_5)
\end{aligned}$$

$$\widehat{\phi}_{rcb} = \phi^1_{rcb} \wedge \phi^2_{rcb} \wedge \phi^3_{rcb} \wedge \phi^4_{rcb} \wedge \phi^5_{rcb} \wedge \phi^6_{rcb}.$$

The Boolean formula $\widehat{\phi}_{rcb}$ is satisfied only with the assignment set $\{00000, 00001, 11000, 11001, 00111\} = \{d_1, d_2, d_3, d_4, d_5\} = \mathcal{D}$.

6.1 Unsatisfiable CNF Formula and No Generable Detectors

In this section, we use our obtained transformation result (Proposition 1) to demonstrate involving properties on the number of detectors that can be generated. An example is the question: Given $\mathcal{S}$ and r, is it possible to generate any detectors at all?

One approach to answer this question is to apply a variant of the Lovász Local Lemma [40]. More specifically we define according to [40], $vbl(C)$ as the set of variables that occur in clause C, i.e. $\{x \in V | x \in C$ or $\overline{x} \in C\}$, where V is a set of Boolean variables. Moreover, as defined in [40], the *neighborhood* of C in ϕ_{rcb} is the set of clauses distinct from C in ϕ_{rcb} that depend on C, or more formally:

$$\begin{aligned}
\Gamma_{\phi_{rcb}}(C) := \{C' \in \phi_{rcb} \,|\, C' \neq C \,\text{and} \\
vbl(C) \cap vbl(C') \neq \emptyset\}
\end{aligned}$$

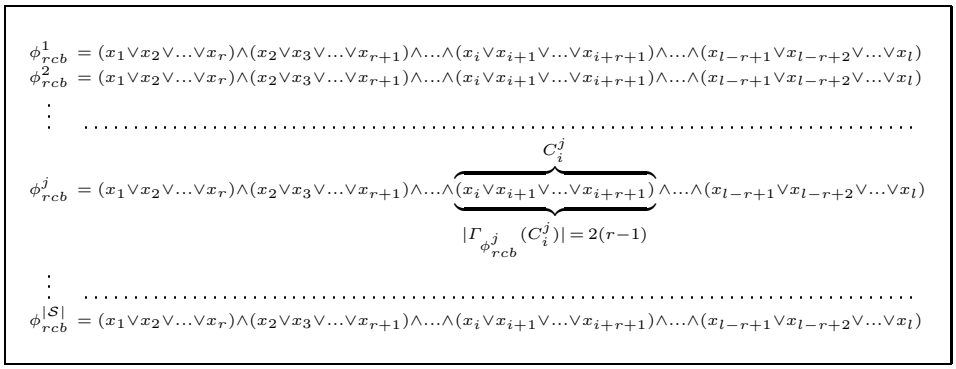


Fig. 10 C^j_i has at most $2\,(r-1)$ many neighborhood clauses in ϕ^j_{rcb} ($r-1$ to left and $r-1$ to right) and at most $(2\,(r-1)+1)\cdot(|\mathcal{S}|-1)$ many neighborhood clauses in all remaining Boolean formulas $\phi^1_{rcb}, \phi^2_{rcb}, \ldots, \phi^{j-1}_{rcb}, \phi^{j+1}_{rcb}, \ldots, \phi^{|\mathcal{S}|}_{rcb}$.

Proposition 2 *Let $\mathcal{S}$ be a set of bit strings of length l, where all $s \in \mathcal{S}$ are consisting of pairwise distinct substrings $s[1,\ldots,r], s[2,\ldots,r+1], \ldots, s[l-r+1,\ldots,l]$. Detectors can be generated, if*

$$|\mathcal{S}| < \frac{2^r\, e^{-1} + 1}{2r-1}.$$

Proof For each $s \in \mathcal{S}$ construct a Boolean formula ϕ^i_{rcb} in l-k-CNF by $\mathfrak{C}(s,r)$. Construct a related k-CNF Boolean formula $\widehat{\phi}_{rcb} = \phi^1_{rcb} \wedge \phi^2_{rcb} \wedge \ldots \wedge \phi^{|S|}_{rcb}$. Let C^j_i be the i-th clause in ϕ^j_{rcb}, $1 \leq j \leq |S|$. C^j_i has at most $2\,(r-1)$ many neighborhood clauses in ϕ^j_{rcb} and at most $(2\,(r-1)+1)\cdot(|S|-1)$ many neighborhood clauses in all remaining Boolean formulas $\phi^1_{rcb}, \phi^2_{rcb}, \ldots, \phi^{j-1}_{rcb}, \phi^{j+1}_{rcb}, \ldots, \phi^{|S|}_{rcb}$. In total this results in $|S| \cdot (2r-1) - 1$ dependent clauses (see Fig. 10).

A variant of the Lovász Local Lemma [40] implies that if $|\Gamma_F(C)| \leq 2^{k-2}$, $k \in \mathbb{N}$ for all clauses C in a k-CNF formula F, then F is satisfiable. Applying the variant of the Lovász Local Lemma results in

$$|\mathcal{S}| \cdot (2r-1) - 1 \leq 2^{r-2} < 2^r/e.$$

A more computational oriented approach to answer the question: Is it possible to generate any detectors at all? Is to apply the Davis-Logemann-Loveland (DLL[5]) algorithm. The DLL algorithm [11] is based on the elimination rules proposed in [12] and terminates either with result unsatisfiable (empty clause) or satisfiable (empty ϕ). Moreover the DLL algorithm can be used to quantify the computational "hardness" of finding detectors by counting the number of backtracking attempts when evaluating the k-CNF input instance.

To be more precise, the algorithm is a depth-first search technique and uses recursive backtracking for guiding the exploration. The algorithm constructs a decision tree, where assignments of the variables coincide with paths from the root to the leafs. If a path leads to an unsatisfiable result, then the algorithm backs up to a different branch. This recursive search is reiterated until it terminates with a satisfiable or unsatisfiable result. In the worst case the whole decision tree has to be inspected, i.e. it will take

[5] The DLL algorithm is sometimes also called DPL or DPLL algorithm [18, 26].

an exponential number of evaluations — similar to an exhaustive search. However on average the DLL algorithm is much faster because it can prune whole branches from the decision tree without exploring the leaves.

Given a Boolean formula ϕ in CNF, a literal l in ϕ and the reduction function $R(\phi, l)$ that outputs the residual formula of ϕ by:

- removing all the clauses that contain l,
- deleting $\overline{l}$ from all the clauses that contain $\overline{l}$,
- removing both l and $\overline{l}$ from the list of literals.

A clause that contains one literal is called *unit clause*, and a literal l is called *monotone*, if $\overline{l}$ appears in no clause of ϕ. In lines 2-7 the reduction function is applied whenever a

Algorithm 3: Davis-Logemann-Loveland algorithm (DLL($\cdot$))

```
   input  : φ (Boolean formula in CNF)
   output: SATISFIABLE or UNSATISFIABLE
 1 begin
 2   forall unit clauses {l} in φ do
 3     φ ← R(φ, l)
 4     if φ includes empty clause then
 5       return UNSATISFIABLE
 6   forall monotone literals l in φ do
 7     φ ← R(φ, l)
 8   if φ is empty then
 9     return SATISFIABLE
10   choose a literal l in φ
11   if DLL(R(φ, l)) = SATISFIABLE) then
12     return SATISFIABLE
13   if DLL(R(φ, l̄)) = SATISFIABLE) then
14     return SATISFIABLE
15   return UNSATISFIABLE
16 end
```

unit clause or a monotone literal is found. The subsequent recursive call is performed in lines 11, 13 respectively. "Easy" input instances imply that the DLL algorithm requires few backtracking attempts because clauses and literals can be efficiently eliminated by means of $R(\phi, l)$ without executing many subsequent recursive calls. On the other hand, "hard" instances imply that many recursive calls or backtracking attempts are required. In the next section, the terms "easy" and "hard" are clarified. More specifically, it will be shown that parameters $|\mathcal{S}|, l$ and r specify the ratio of the number of clauses to variables of the $\widehat{\phi}_{rcb}$ instances and therefore characterize the computational complexity of the DLL algorithm.

6.2 Phase Transition in k-CNF Satisfiability

The k-CNF satisfiability problem is $\mathcal{NP}$-complete for $k > 2$, however, this fact does not imply that *all* instances of the k-CNF satisfiability problem are intractable to solve. In

point of fact, there exists many problem instances which are "easy" to solve, i.e. one can efficiently decide whether the instance is satisfiable or is unsatisfiable. On the other hand there also exist problem instances which are "hard", i.e. one can *not* efficiently decide whether the instance is satisfiable or is not satisfiable. The computational "hardness" of finding assignments sets for randomly generated instances is characterized by the ratio [20]

$$r_k = \frac{\text{number of clauses}}{\text{number of variables}}. \tag{14}$$

If the Boolean formula ϕ has many variables and few clauses, then ϕ is *under-constrained* and as a result, there exist many assignment sets. The DLL algorithm requires for under-constrained problem instances few backtracking attempts and therefore "easily" deduces the satisfiability. On the other hand, if the ratio of the number of clauses to variables is large, then ϕ is *over-constrained* and almost has no satisfying assignment set. Such over-constrained instances are likewise "easily" deducible for the DLL algorithm. However, there also exists a transition from under-constrained to the over-constrained region. In such a *phase transition* region, the probability of the instances being satisfiable equals 0.5 and thus one has the largest uncertainty whether the instances are satisfiable or are unsatisfiable.

For the 3-CNF satisfiability problem, the ratio (phase transition threshold) is experimentally approximated by 4.24 [18,31]. That means, when r_3 is close[6] to 4.24, the DLL algorithm has to backtrack most frequently to determine the final result. If the Boolean formula is under-constrained ($r_3 < 4.24$) or over-constrained ($r_3 > 4.24$), then the algorithm prunes whole branches from the decision tree, without exploring the leaves and terminates after few recursive calls.

6.3 Average Number of Distinct Clauses

Given $\mathcal{S}, l$ and r, the constructed Boolean formula $\widehat{\phi}_{rcb}$ contains in total $(l-r+1)\cdot|\mathcal{S}|$ clauses. However, $\widehat{\phi}_{rcb}$ does not necessarily contain $(l-r+1)\cdot|\mathcal{S}|$ pairwise distinct clauses. Two clauses are distinct from each other, if they differ in at least one literal. In point of fact, if $l \gg r$, then a large number of equal clauses occur in $\widehat{\phi}_{rcb}$. Equal clauses in $\widehat{\phi}_{rcb}$ however, do not bias the computational complexity. To quantify the computational complexity by means of the DLL algorithm, one has to determine the average number of pairwise distinct clauses.

Example 4 Let $\mathcal{S} := \{0101, 1101\}$ and $r = 3$, hence $\widehat{\phi}_{rcb}$ results in

$$(x_1 \vee \overline{x}_2 \vee x_3) \wedge (\overline{x}_2 \vee x_3 \vee \overline{x}_4) \wedge (\overline{x}_1 \vee \overline{x}_2 \vee x_3) \wedge (\overline{x}_2 \vee x_3 \vee \overline{x}_4).$$

Example 4 shows that the second and the fourth clause are equal, because the last three bits of 0101 and 1101 are equal.

Proposition 3 *Given bit string length l, matching length r and $\mathcal{S}$ which contains pairwise distinct bit strings $s_1, s_2, \ldots, s_{|\mathcal{S}|}$ randomly drawn from $\mathcal{U}$. The average number of pairwise distinct clauses is*

$$\mathbf{E}[|\widehat{\phi}_{rcb}|] = 2^r\,(l-r+1) - \left(1 - \frac{1}{(l-r+1)\,2^r}\right)^{|\mathcal{S}|(l-r+1)} (l-r+1)\,2^r. \tag{15}$$

[6] It is still an open problem to prove where the *exact* phase transition threshold is located. Latest theoretical work [1] shows that the threshold r_k lies within the boundary $2.68 < r_k < 4.51$ for $k = 3$.

Proof Construct a lookup table $\mathfrak{T}$ which contains all $2^r \cdot (l-r+1)$ clauses with label T and is of the form

clause	*label*
$(x_1 \vee x_2 \vee \ldots \vee x_{r-1} \vee x_r)$	T
$(x_2 \vee x_3 \vee \ldots \vee x_r \vee x_{r+1})$	T
$\vdots$	$\vdots$
$(x_{l-r+1} \vee x_{l-r+2} \vee \ldots \vee x_{l-1} \vee x_l)$	T
$(x_1 \vee x_2 \vee \ldots \vee x_{r-1} \vee \overline{x}_r)$	T
$(x_2 \vee x_3 \vee \ldots \vee x_r \vee \vee \overline{x}_{r+1})$	T
$\vdots$	$\vdots$
$(x_{l-r+1} \vee x_{l-r+2} \vee \ldots \vee x_{l-1} \vee \overline{x}_l)$	T
$\vdots$	$\vdots$
$(\overline{x}_1 \vee \overline{x}_2 \vee \ldots \vee \overline{x}_{r-1} \vee \overline{x}_r)$	T
$(\overline{x}_2 \vee \overline{x}_3 \vee \ldots \vee \overline{x}_r \vee \overline{x}_{r+1})$	T
$\vdots$	$\vdots$
$(\overline{x}_{l-r+1} \vee \overline{x}_{l-r+2} \vee \ldots \vee \overline{x}_{l-1} \vee \overline{x}_l)$	T

Transform $\mathcal{S}$ into the corresponding Boolean formula $\widehat{\phi}_{rcb}$ and set the label to F whenever a clause in $\mathfrak{T}$ is member of $\widehat{\phi}_{rcb}$. As $\mathcal{S}$ is randomly drawn without replacement from $\mathcal{U}$, the F and T labels are binomially distributed in $\mathfrak{T}$. The probability of finding *no* clauses which are labeled with F when randomly drawn $|\mathcal{S}| \cdot (l-r+1)$ clauses from $\mathfrak{T}$ results in

$$\left(1 - \frac{1}{(l-r+1)\,2^r}\right)^{|\mathcal{S}|(l-r+1)}$$

and hence, the total number of clauses with label F results in

$$2^r\,(l-r+1) - \left(1 - \frac{1}{(l-r+1)\,2^r}\right)^{|\mathcal{S}|(l-r+1)} (l-r+1)\,2^r.$$

7 Experiment with $\widehat{\phi}_{rcb}$ Instances

The computational complexity of finding detectors is experimentally investigated with the DLL algorithm. More specifically, the parameters $l = 75, r = 3$ are chosen and $|\mathcal{S}|$ is varied from 1 to 25, i.e. for each cardinality value from 1 to 25, $\mathcal{S}$ contains distinct bit strings which are randomly drawn from $\mathcal{U}$. As a result, one obtains Boolean formulas $\widehat{\phi}_{rcb}$ in 75-3-CNF with 75 variables and $(75-3+1) \cdot |\mathcal{S}|$ clauses, $\mathbf{E}[|\widehat{\phi}_{rcb}|]$ distinct clauses, respectively. To obtain a large number of different $\widehat{\phi}_{rcb}$ instances, for each value of $|\mathcal{S}|$, 300 instances are randomly generated. The DLL algorithm is applied on each instance and the results: satisfiable/unsatisfiable and the number of backtracking attempts are noted. The result is depicted in figure 11. The abscissa denotes the ratio of the average number of distinct clauses to variables. The ordinate denotes the number of backtracking attempts (computational costs). The resulting ordinate values are colored gray if the DLL algorithm outputs satisfiable, otherwise it outputs unsatisfiable and the values are colored black. One can see in figure 11 that for $(r_3 < 4)$ a large number of satisfiable instances exist. Or to say it the other way around,

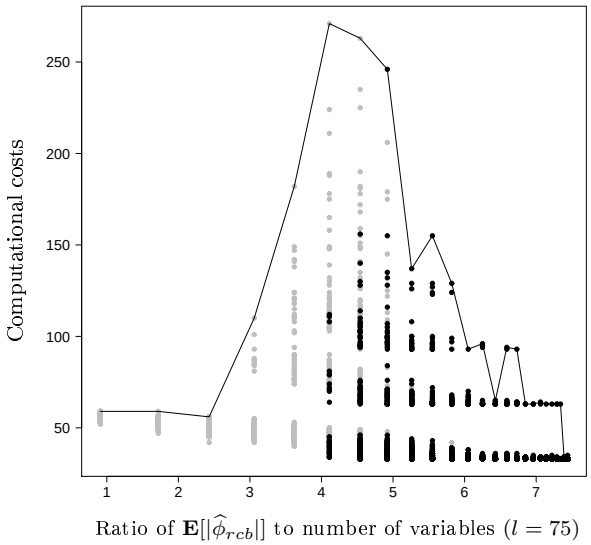


Fig. 11 Number of backtracking attempts (computational costs) of the DLL algorithm to decide whether a $\widehat{\phi}_{rcb}$ instance is satisfiable or unsatisfiable. The gray points denote satisfiable instances whereas black points denote unsatisfiable instances. The "hardest" instances are lying in the interval 4 to 5, termed phase transition region.

for small values of $|\mathcal{S}|$ the resulting Boolean formula $\widehat{\phi}_{rcb}$ is under-constrained and therefore a large number of satisfiable instances exist. The DLL algorithm hence "easily" deduces a satisfiability result. The number of satisfiable and unsatisfiable instances is nearly equal for $(4 < r_3 < 5)$. These instances have the largest uncertainty for the DLL algorithm. As a consequence, the DLL algorithm requires the most backtracking attempts to determine whether the instances are satisfiable or are unsatisfiable. A ratio $(r_3 > 5)$ implies that a large number of over-constrained instances exist and hence, the DLL algorithm "easily" deduces the unsatisfiable result. Another way to visualize this "easy-hard-easy" pattern, is to plot the percentage of satisfiable instances on the ordinate (see Fig. 12). One can see that the probability of the instances being satisfiable equals 0.5 when $(4 < r_3 < 5)$ and rapidly changes to 1 for $(r_3 < 4)$ and to 0 for $(r_3 > 5)$.

7.1 Complexity of Algorithms to Generate Detectors

In the last 10 years several algorithms are proposed to generate detectors. Moreover, it was an open problem whether generating *all* detectors can be performed efficiently, i.e. in polynomial time and with polynomial space occupation with regard to parameters r and $|\mathcal{S}|$, because all proposed algorithms (see Fig 7.1), either have a time or a space complexity which is exponential in the matching length r, i.e. $\mathcal{O}(2^r)$.

There seems to be strong evidence that finding *all* detectors require at least $\Omega(2^r)$ bit string evaluations. This assumption is thereby justified, that $\Omega(2^r)$ evaluations are required for finding all satisfying sets for the first clause of each $s \in \mathcal{S}$. Additionally,

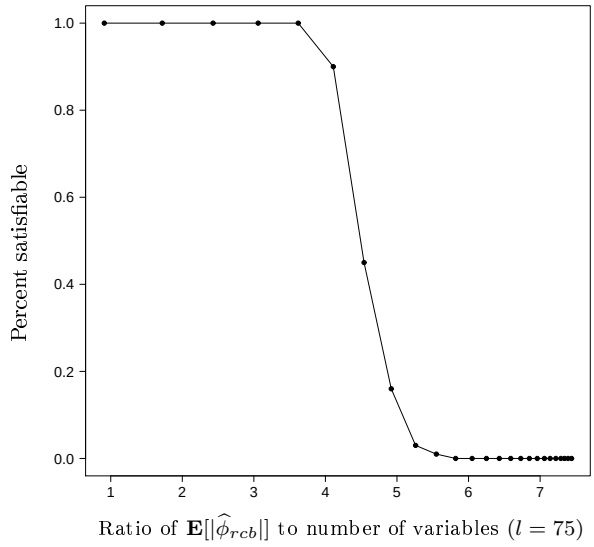


Fig. 12 Coherence between the percentage of satisfiable instances and the ratio of $\mathbf{E}[|\widehat{\phi}_{rcb}|]/l$. The "hardest" instances live in the region where the number of satisfiable and unsatisfiable instances is equal, or in other words, the probability of instances being satisfiable equals 0.5.

Linear time detector generating algorithm [13]:

Time : $\mathcal{O}\left((l-r)\cdot|\mathcal{S}|\right)+\mathcal{O}\left((l-r)\cdot 2^r\right)+\mathcal{O}\left(l\cdot|\mathcal{D}|\right)$
Space : $\mathcal{O}\left((l-r)^2\cdot 2^r\right)$

Greedy detector generating algorithm [13]:

Time : $\mathcal{O}\left((l-r)\cdot|\mathcal{D}|\cdot 2^r\right)$
Space : $\mathcal{O}\left((l-r)^2\cdot 2^r\right)$

Binary template algorithm [42]:

Time : $\mathcal{O}\left((l-r)\cdot 2^r\cdot|\mathcal{D}|\right)+\mathcal{O}\left(2^r\cdot|\mathcal{S}|\right)$
Space : $\mathcal{O}\left((l-r)\cdot 2^r\right)+\mathcal{O}\left(|\mathcal{D}|\right)$

NSMutation algorithm [2]:

Time : $\mathcal{O}\left(2^l\cdot|\mathcal{S}|\right)+\mathcal{O}\left(|\mathcal{D}|\cdot 2^r\right)+\mathcal{O}\left(|\mathcal{D}|\right)$
Space : $\mathcal{O}\left(l\cdot(|\mathcal{S}|+|\mathcal{D}|)\right)+\mathcal{O}(|\mathcal{D}|)$

Fig. 13 Complexity overview of proposed algorithms for generating detectors.

the remaining $(l-r)$ clauses of each $s \in \mathcal{S}$ must be verified, which in total could be done in at most $\mathcal{O}(|\mathcal{S}| \cdot 2^k)$ evaluations. Moreover as outlined in section 6, the k-CNF satisfiability problem is a decision problem, where the input is a Boolean formula f and the output is "Yes", if f is satisfiable, and "No", otherwise. The currently fastest known deterministic algorithm that decides the 3-CNF problem, runs in time $\mathcal{O}(1.473^n)$ [5], where n is the number of variables. The probabilistic algorithm variant runs in time $\mathcal{O}(1.3302^n)$ [23].

We would like to emphasize here, that the deterministic and probabilistic k-CNF algorithms decide if a Boolean formula is satisfiable. However the algorithms do not determine *all* satisfiable assignment sets — in our case, all detectors that can be generated.

8 Conclusion

Negative selection and the associated r-contiguous matching rule is a popular immune-inspired method for anomaly detection problems. However, until now there has been limited available theoretical work from the perspective of a pure pattern classification problem and the computational complexity of generating detectors. In this article we studied the generalization capability of negative selection and the associated r-contiguous matching rule. Moreover, the partition of the universe $\mathcal{U}$ in distinct subsets $\mathcal{S}, \mathcal{H}$ and $\mathcal{N}$ was explained. Based on the probability of the r-contiguous matching rule, the average number of detectors and holes was determined and the resulted implications were discussed. Furthermore, we have shown that the problem of generating r-contiguous detectors, when given $\mathcal{S}$ and matching length r can be transformed into an instance of the k-CNF satisfiability problem. The assignment set of the Boolean formula in k-CNF is the detector set $\mathcal{D}$. This result was exploited to provide insights into the computational complexity of generating detectors and insights when detectors can be generated at all.

Based on these observed facts we can state that the negative selection algorithm and the associated r-contiguous matching rule is *not appropriate* for anomaly detection problems.

Acknowledgment

The author thanks Erin Gardner and Dawn Yackzan for their valuable suggestions and comments.

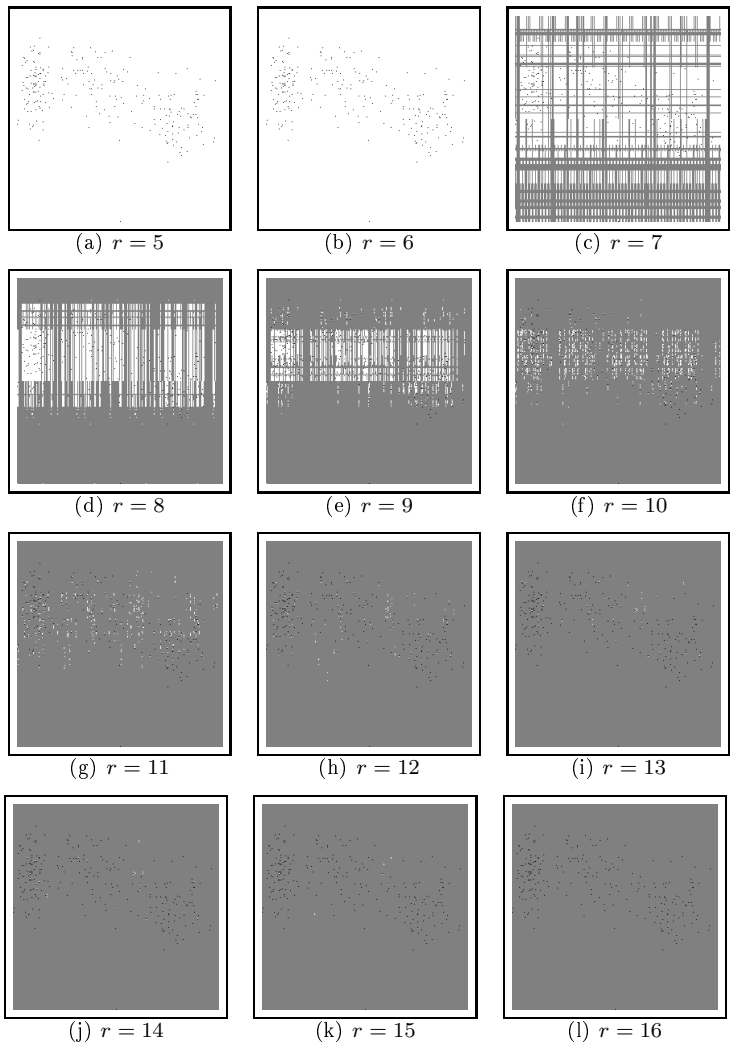
References

1. Achlioptas, D., Naor, A., Peres, Y.: Rigorous location of phase transitions in hard optimization problems. Nature **435**, 759–764 (2005)
2. Ayara, M., Timmis, J., de Lemos, R., de Castro, L.N., Duncan, R.: Negative selection: How to generate detectors. In: Proceedings of the 1nd International Conference on Artificial Immune Systems (ICARIS), pp. 89–98. University of Kent at Canterbury Printing Unit (2002)
3. Balthrop, J., Forrest, S., Glickman, M.: Revisiting lisys: Parameters and normal behavior. In: Proceedings of Congress On Evolutionary Computation (CEC), pp. 1045–1050. IEEE Press (2002)

4. Bishop, C.M.: Novelty detection and neural network validation. IEE Proceedings - Vision, Image and Signal processing **141**(4), 217–222 (1994)
5. Brueggemann, T., Kern, W.: An improved deterministic local search algorithm for 3-SAT. Theoretical Computer Science **329**(1–3), 303–313 (2004)
6. de Castro, L.N., Timmis, J.: Artificial Immune Systems: A New Computational Intelligence Approach. Springer Verlag (2002)
7. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, second edn. MIT Press (2002)
8. Dasgupta, D., Forrest, S.: Tool breakage detection in milling operations using a negative-selection algorithm. Tech. Rep. CS95-5, University of New Mexico (1995)
9. Dasgupta, D., Forrest, S.: Novelty detection in time series data using ideas from immunology. In: Proceedings of the 5th International Conference on Intelligent Systems, pp. 82–87 (1996)
10. Dasgupta, D., Forrest, S.: Artificial Immune Systems and their Applications, chap. An Anomaly Detection Algorithm Inspired by the Immune System, pp. 262–277. Springer-Verlag (1998)
11. Davis, M., Logemann, G., Loveland, D.: A machine program for theorem-proving. Communications of the ACM **5**(7), 394–397 (1962)
12. Davis, M., Putnam, H.: A computing procedure for quantification theory. Journal of the ACM (JACM) **7**(3), 201–215 (1960)
13. D'haeseleer, P., Forrest, S., Helman, P.: An immunological approach to change detection: algorithms, analysis, and implications. In: Proceedings of the Symposium on Research in Security and Privacy, pp. 110–119. IEEE Computer Society Press (1996)
14. Esponda, F., Forrest, S.: Detector coverage under the r-contiguous bits matching rule. Tech. Rep. TR-CS-2002-03, University of New Mexico (2002)
15. Esponda, F., Forrest, S., Helman, P.: The crossover closure and partial match detection. In: Proceedings of the 2nd International Conference on Artificial Immune Systems (ICARIS), *Lecture Notes in Computer Science*, vol. 2787, pp. 249–260. Springer-Verlag (2003)
16. Feller, W.: An Introduction to Probability Theory and its Applications, vol. 1, 3. edn. John Wiley & Sons (1968)
17. Forrest, S., Perelson, A.S., Allen, L., Cherukuri, R.: Self-nonself discrimination in a computer. In: Proceedings of the Symposium on Research in Security and Privacy, pp. 202–212. IEEE Computer Society Press (1994)
18. Freeman, J.W.: Hard random 3-SAT problems and the Davis-Putnam procedure. Artificial Intelligence **81**(1–2), 183–198 (1996)
19. Freitas, A.A., Timmis, J.: Revisiting the foundations of artificial immune systems: A problem-oriented perspective. In: Proceedings of the 2nd International Conference on Artificial Immune Systems (ICARIS), *Lecture Notes in Computer Science*, vol. 2787, pp. 229–241. Springer-Verlag (2003)
20. Gent, I.P., Walsh, T.: The SAT phase transition. In: Proceedings of the 11th European Conference on Artificial Intelligence, pp. 105–109. John Wiley & Sons (1994)
21. Glickman, M., Balthrop, J., Forrest, S.: A machine learning evaluation of an artificial immune system. Evolutionary Computation **13**(2), 179–212 (2005)
22. González, F., Dasgupta, D., Gómez, J.: The effect of binary matching rules in negative selection. In: Genetic and Evolutionary Computation — GECCO, *Lecture Notes in Computer Science*, vol. 2723, pp. 195–206. Springer-Verlag, Chicago (2003)
23. Hofmeister, T., Schöning, U., Schuler, R., Watanabe, O.: A probabilistic 3-SAT algorithm further improved. In: 19th Annual Symposium on Theoretical Aspects of Computer Science (STACS), *Lecture Notes in Computer Science*, vol. 2285, pp. 192–202. Springer-Verlag (2002)
24. Hofmeyr, S.A.: An immunological model of distributed detection and its application to computer security. Ph.D. thesis, University of New Mexico (1999)
25. Mitchell, T.: Machine Learning. McGraw Hill (1997)
26. Ouyang, M.: How good are branching rules in DPLL. Discrete Applied Mathematics **89**(1-3), 281–286 (1998)
27. Percus, J.K., Percus, O.E., Perelson, A.S.: Predicting the size of the T-cell receptor and antibody combining region from consideration of efficient self-nonself discrimination. Proceedings of National Academy of Sciences USA **90**, 1691–1695 (1993)
28. Ranang, M.T.: An artificial immune system approach to preserving security in computer networks. Master's thesis, Norges Teknisk-Naturvitenskapelige Universitet (2002)
29. Reischuk, K.R.: Einführung in die Komplexitätstheorie. B.G. Teubner Stuttgart (1990)

30. Schölkopf, B., Platt, J.C., Shawe-Taylor, J., Smola, A.J., Williamson, R.C.: Estimating the support of a high-dimensional distribution. Neural Computation **13**(7), 1443–1471 (2001)
31. Selman, B., Mitchell, D.G., Levesque, H.J.: Generating hard satisfiability problems. Artificial Intelligence **81**(1–2), 17–29 (1996)
32. Singh, S.: Anomaly detection using negative selection based on the r-contiguous matching rule. In: Proceedings of the 1st International Conference on Artificial Immune Systems (ICARIS), pp. 99–106. Unversity of Kent at Canterbury Printing Unit (2002)
33. Steinwart, I., Hush, D., Scovel, C.: A classification framework for anomaly detection. Journal of Machine Learning Research **6**, 211–232 (2005)
34. Stibor, T., Timmis, J., Eckert, C.: Generalization regions in hamming negative selection. In: Intelligent Information Processing and Web Mining, Advances in Soft Computing, pp. 447–456. Springer-Verlag (2006)
35. Stibor, T., Timmis, J., Eckert, C.: The link between r-contiguous detectors and k-CNF satisfiability. In: Proceedings of Congress On Evolutionary Computation (CEC), pp. 491–498. IEEE Press (2006)
36. Tarassenko, L., Hayton, P., Cerneaz, N., Brady, M.: Novelty detection for the identification of masses in mammograms. In: Proceedings of the 4th IEE International Conference on Artificial Neural Networks, pp. 442–447 (1995)
37. Tax, D.M.J., Duin, R.P.W.: Data domain description using support vectors. In: European Symposium on Artificial Neural Networks – ESANN, pp. 251–256 (1999)
38. Taylor, D.W., Corne, D.W.: An investigation of the negative selection algorithm for fault detection in refrigeration systems. In: Proceedings of the 2nd International Conference on Artificial Immune Systems (ICARIS), *Lecture Notes in Computer Science*, vol. 2787, pp. 34–45. Springer-Verlag (2003)
39. Uspensky, J.V.: Introduction to Mathematical Probability. McGraw-Hill (1937)
40. Welzl, E.: Boolean satisfiability — combinatorics and algorithms (2005). Lecture Notes (http://www.inf.ethz.ch/∼emo/SmallPieces/SAT.ps)
41. Wierzchoń, S.T.: Discriminative power of the receptors activated by k-contiguous bits rule. Journal of Computer Science and Technology **1**(3), 1–13 (2000)
42. Wierzchoń, S.T.: Generating optimal repertoire of antibody strings in an artificial immune system. In: Intelligent Information Systems, pp. 119–133. Springer Verlag (2000)

9 Appendix

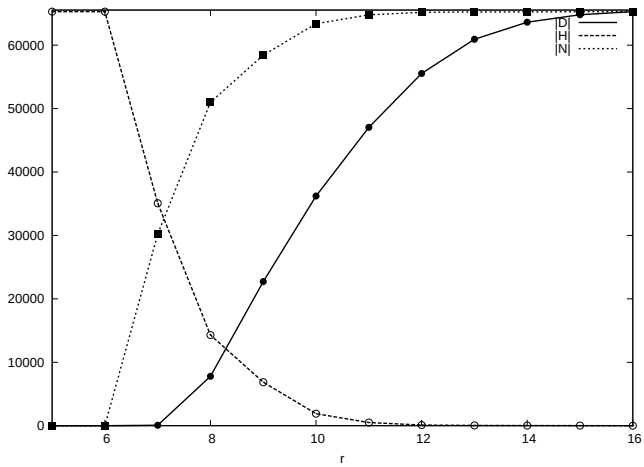


(a) $r = 5$ (b) $r = 6$ (c) $r = 7$

(d) $r = 8$ (e) $r = 9$ (f) $r = 10$

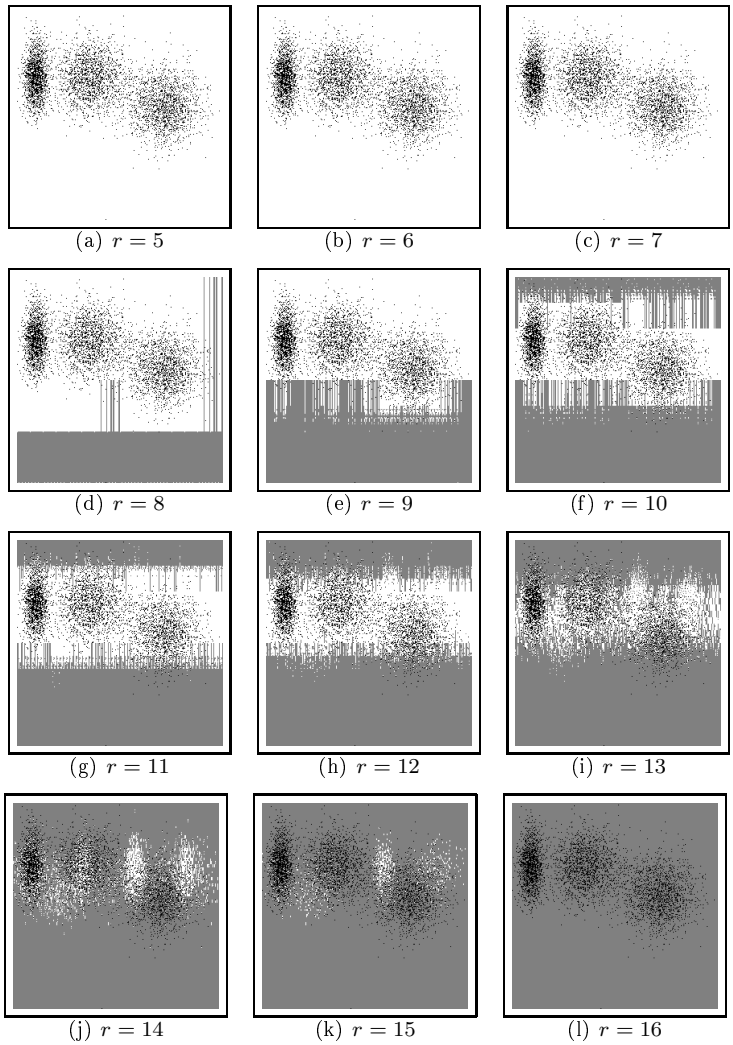
(g) $r = 11$ (h) $r = 12$ (i) $r = 13$

(j) $r = 14$ (k) $r = 15$ (l) $r = 16$



(m) Coherence between $|\mathcal{D}|, |\mathcal{H}|$ and $|\mathcal{N}|$ of the above depicted figures 14(a)-14(l).

Fig. 14 Coherence between detector coverage and induced holes for stepwise increasing matching length r. The gray shaded area is covered by the generated detectors, the white area represents holes. The black points represent normal examples ($|\mathcal{S}| = 250$) which are generated by a mixture of Gaussian distributions (see Fig. 6(a)).

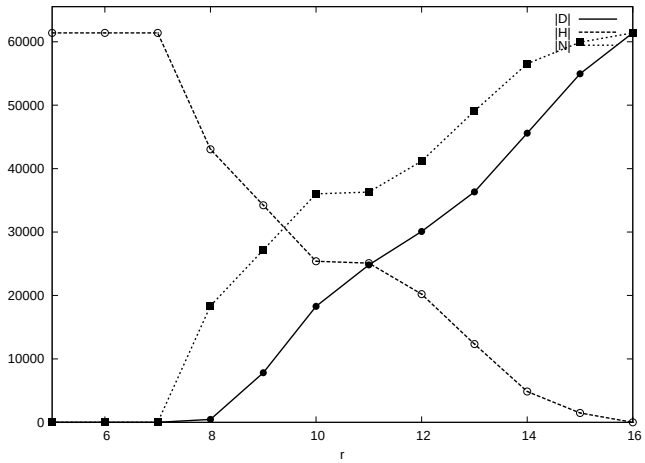


(a) $r = 5$ (b) $r = 6$ (c) $r = 7$

(d) $r = 8$ (e) $r = 9$ (f) $r = 10$

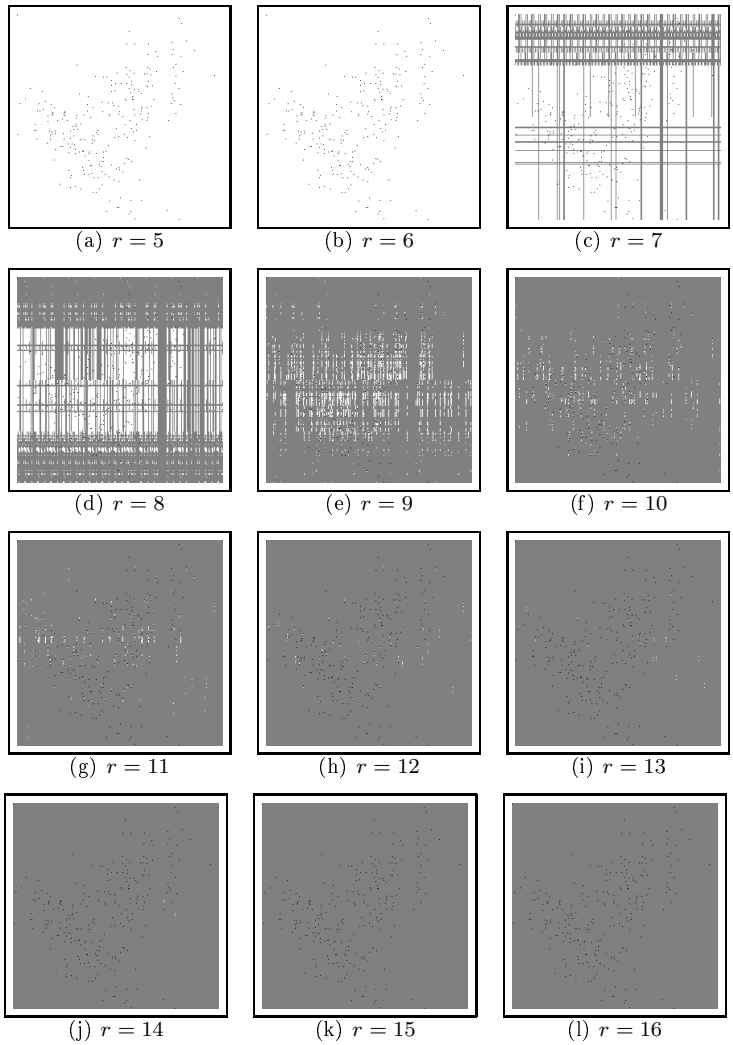
(g) $r = 11$ (h) $r = 12$ (i) $r = 13$

(j) $r = 14$ (k) $r = 15$ (l) $r = 16$



(m) Coherence between $|\mathcal{D}|, |\mathcal{H}|$ and $|\mathcal{N}|$ of the above depicted figures 15(a)-15(l).

Fig. 15 Visualized experimental results as in Fig. 14, however with $|\mathcal{S}| = 5000$ generated normal examples.

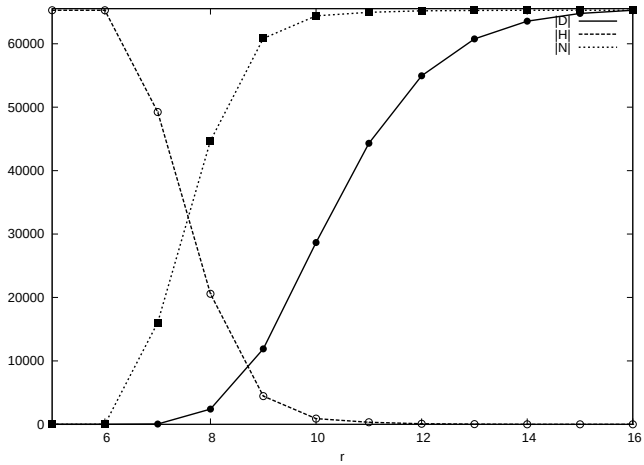


(a) $r = 5$ (b) $r = 6$ (c) $r = 7$

(d) $r = 8$ (e) $r = 9$ (f) $r = 10$

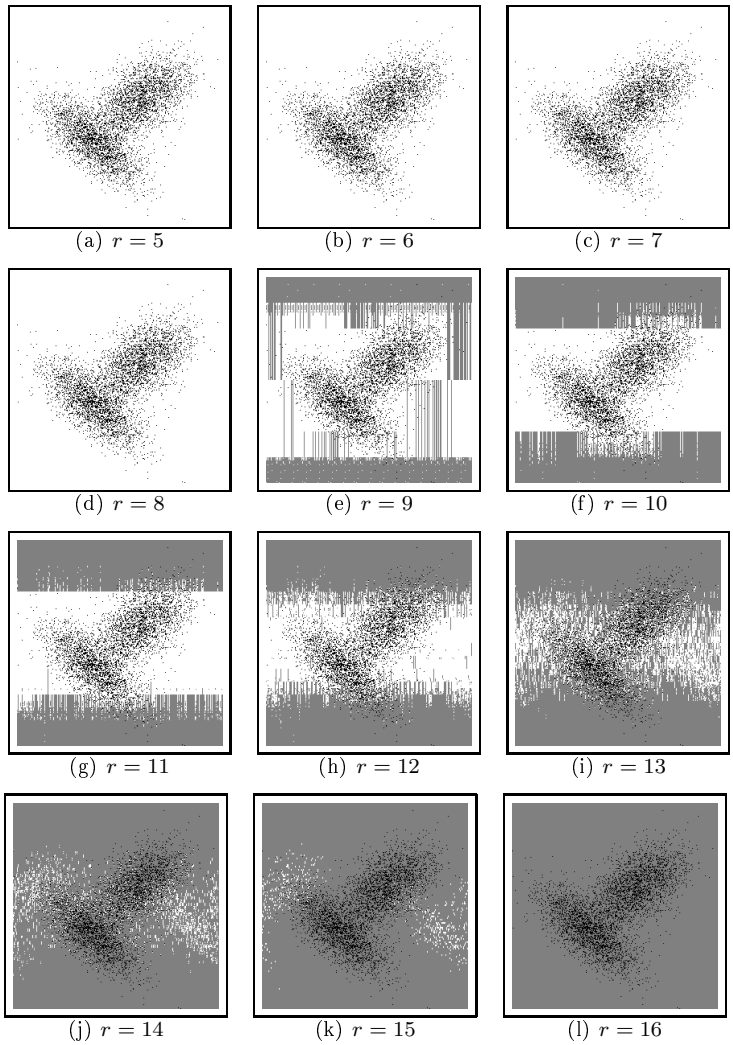
(g) $r = 11$ (h) $r = 12$ (i) $r = 13$

(j) $r = 14$ (k) $r = 15$ (l) $r = 16$



(m) Coherence between $|\mathcal{D}|, |\mathcal{H}|$ and $|\mathcal{N}|$ of the above depicted figures 16(a)-16(l).

Fig. 16 Visualized experimental results as in Fig. 14, however the $|\mathcal{S}| = 250$ generated normal examples are sampled of probability distribution depicted in Fig. 6(b).

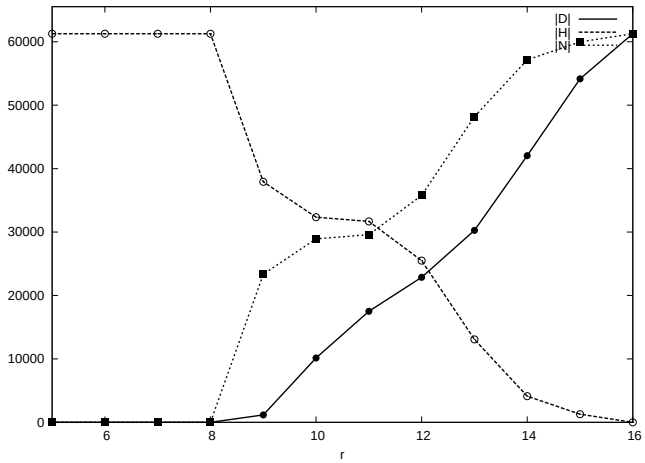


(a) $r = 5$ (b) $r = 6$ (c) $r = 7$

(d) $r = 8$ (e) $r = 9$ (f) $r = 10$

(g) $r = 11$ (h) $r = 12$ (i) $r = 13$

(j) $r = 14$ (k) $r = 15$ (l) $r = 16$



(m) Coherence between $|\mathcal{D}|, |\mathcal{H}|$ and $|\mathcal{N}|$ of the above depicted figures 17(a)-17(l).

Fig. 17 Visualized experimental results as in Fig. 16, however with $|\mathcal{S}| = 5000$ generated normal examples.