Efficient Negative Selection Algorithms by Sampling and Approximate Counting

Johannes Textor

Theoretical Biology & Bioinformatics
Universiteit Utrecht
Paudalaan 8
3584 CH Utrecht, NL
johannes.textor@gmx.de

Abstract. Negative selection algorithms (NSAs) are immune-inspired anomaly detection schemes that are trained on normal data only: A set of *consistent detectors* – i.e., detectors that do not match any element of the training data – is generated by rejection sampling. Then, input elements that are matched by the generated detectors are classified as anomalous. NSAs generally suffer from exponential runtime. Here, we investigate the possibility to accelerate NSAs by sampling directly from the set of consistent detectors. We identify conditions under which this approach yields fully polynomial time randomized approximation schemes of NSAs with exponentially large detector sets. Furthermore, we prove that there exist detector types for which the approach is feasible even though the only other known method for implementing NSAs in polynomial time fails. These results provide a firm theoretical starting point for implementing efficient NSAs based on modern probabilistic techniques like Markov Chain Monte Carlo approaches.

1 Introduction

The adaptive immune system is, alongside the central nervous system, one of the two important cognitive systems in vertebrates. Within this system, the *T cells* are responsible for performing many important cognitive tasks, like detecting viral infections in cells. Because T cells can perform actions with potentially hazardous consequences for their host organism – e.g., they can kill cells that express "anomalous" surface molecules – it is important to ensure that T cells do not incorrectly classify normal metabolic activity as anomalous. At the same time, it is crucial that an organism's T cell repertoire provides protection against the huge set of pathogens it may potentially encounter. *Negative selection* is an immunological process that helps achieve these two goals: Newborn T cells with randomly generated receptors are exposed to normal molecular structures from the host organism (self), and those that react to any self structure are killed. Only cells that survive negative selection become part of the T cell repertoire. Motivated by a need for better computer security systems, Forrest et al. [1] conceived a generic classification scheme, which they called the *negative selection algorithm* (NSA), that mimics this simple yet effective biological paradigm.

C.A. Coello Coello et al. (Eds.): PPSN 2012, Part I, LNCS 7491, pp. 32–41, 2012.

The NSA scheme can be applied to very diverse types of data. We will assume that the data to be classified originates from a universe $\mathcal{U}$, which is usually parameterized by some index L that characterizes the length of an input element (e.g., for an alphabet Σ, we might use $\mathcal{U} = \Sigma^L$). Moreover, we assume that a basis set of patterns (detectors) $\mathcal{P}$ is given, such that each $\pi \in \mathcal{P}$ matches a subset of $\mathcal{U}$. The patterns in $\mathcal{P}$ represent T cells, while the elements of $\mathcal{U}$ represent the molecular structures examined by T cells. For instance, a frequently used class of patterns, motivated by a model of how real T cells "see" antigen [2], are the so-called "r-contiguous detectors". Here, $\mathcal{U} = \mathcal{P} = \Sigma^L$, and a pattern is said to match a universe element if both are identical in at least r contiguous positions. E.g., for $r = 2$ and $\mathcal{U} = \{0,1\}^3$, the patterns 011 and 110 match the string 010 but the pattern 111 does not. As another example, we could use $\mathcal{U} = \{0,1\}^L$ and $\mathcal{P} = \{0,1,*\}^L$ to encode binary patterns with don't-care-symbols (e.g. $0**$ would match all strings of length 4 starting with 0).

```
NegativeSelection(S, M, n).
Input: Sample S ⊆ U, set M ⊆ U, integer n.
Output: For each m ∈ M, either (m, +1) or (m, −1).
1   D ← ∅
2   while |D| < n do        // training step
3       pick π ∈ P uniformly at random
4       if π does not match any s ∈ S then
5           D ← D ∪ {π}
6   for each m ∈ M do    // classification step
7       if any π ∈ D matches m then
8           output (m, +1)
9       else
10          output (m, −1)
```

Fig. 1. Pseudocode of the negative selection algorithm (NSA) considered in this paper. In the literature on NSAs (e.g., [3,4]), S is frequently called a *self set*, M a *monitor set*, and D is called the *detector set*. For the sake of conciseness, we treat the set D as a multi-set, i.e., D can contain the same detector more than once.

Fig. 1 shows the pseudocode for a typical NSA scheme. A set of detectors D with size n is generated by rejection sampling, i.e., detectors are sampled at uniform and added to D if they match no element of the input sample S. Subsequently, in the classification step, the elements of M are classified as anomalous $(+1)$ if matched by any detector in D and as normal (-1), otherwise.

2 Related Work and Our Contribution

Layman implementations of the NSA typically suffer from exponential runtime [5,4]. Two issues can arise: First, the rejection sampling step may be rate-limiting if the input set S is "large" such that most detectors match some $s \in S$, and have to be rejected. Second, prohibitively many detectors might be needed to achieve

acceptable detection rates in the classification step. The main contribution of this work is that we establish a novel possibility of implementing NSA algorithms efficiently: We investigate under which conditions it is possible to replace the rejection sampling in the training step by a more efficient procedure that requires only polynomial time to find a single detector. Moreover, we show that a similar sampling approach can be employed to determine the NSA outcome approximately even when n is very large.

Our research on the efficiency of NSAs has two main motivations. First, a widely held view in the field of artificial immune systems (AIS) used to be that NSAs cannot be efficiently implemented. E.g., for r-contiguous detectors, it was hypothesized that even deciding the existence of a single detector that fails to match all $s \in S$ is already NP-complete [3]. This idea led some researchers to conclusions like "negative selection [algorithms] ... can never scale" [6], or that "future work in this direction is not meaningful" [5]. It stands to reason that one wishes to verify whether unproved claims that motivate such bold statements really hold true. For some special cases, we have previously shown that polynomial-time NSAs can be obtained using the "detector compression" technique [7,8], thus disproving the NP-completeness hypothesis for r-contiguous detectors [3]. However, we subsequently proved that the detector compression technique is not applicable to many interesting types of detectors [9], which raised the question whether other methods might exist to obtain efficient NSAs. The technique that we put forward in this paper is able to address at least some cases where detector compression is infeasible.

Independently of the debate in the AIS community, NSAs find their main application in the field of theoretical immunology, where they are used as components of simulations of the real immune system (e.g. [10,11,12]). Recently, a NSA-based model was used to show that genetically determined differences in negative selection can partly explain why certain individuals are able to control HIV infections [13]. In these computational biology applications, the NSA cannot be replaced by traditional machine learning methods because it is used as a simulation of the real negative selection process rather than merely a generic classification scheme.

Therefore, increasing the efficiency of NSAs benefits not only AIS but is also important for computational immunology.

3 Our Approach

The basic idea behind our approach is very simple: Instead of generating the detector set D in the training step by rejection sampling (lines 2-5 in Fig. 1), we sample directly from the subset of those detectors in $\mathcal{P}$ that do not match any $s \in S$. For instance, in many cases we might be able to construct a graph that encodes the desired detectors, perform a "sufficiently long" random walk on this graph, and output the last vertex we visit. Provided the random walk is "rapidly mixing" (i.e., approaches the equilibrium distribution sufficiently fast),

this approach, which is known as the *Markov Chain Monte Carlo method* [14], can efficiently generate an approximately uniform sample from the set of S-consistent detectors.

To proceed, we need some notation. We abbreviate the set $\{1, \ldots, n\} \subseteq \mathbb{N}$ by $[1, n]$ and the set $\{-1, 1\}$ by ± 1. Moreover we write "u.a.r." instead of "uniformly at random". Given a universe $\mathcal{U}$, a *detector type* $\mathcal{D} = (\mathcal{P}, \mathcal{M})$ is a tuple of a set $\mathcal{P}$ of *patterns* (or detectors) and a *matching function* $\mathcal{M} : \mathcal{P} \times \mathcal{U} \to \pm 1$. Given a detector $\pi \in \mathcal{P}$ and an element $x \in \mathcal{U}$, if $\mathcal{M}(\pi, x) = 1$ we say "π matches x" and "π does not match x", otherwise. A *sample* is a labeled set $S \subseteq \mathcal{U} \times \pm 1$. A *negative sample* is a sample in which all labels are -1. A detector $\pi \in \mathcal{P}$ is called S-*consistent* if for every $(x, +1) \in S$, we have $\mathcal{M}(\pi, x) = +1$, and for every $(x, -1) \in S$, we have $\mathcal{M}(\pi, x) = -1$. A detector set $D \subseteq \mathcal{P}$ is called S-consistent if it only contains S-consistent detectors. The set of *all* S-consistent detectors is written as $\mathcal{D}[S]$. The *consistency problem* is defined as follows: Given a sample S, decide whether $\mathcal{D}[S]$ is empty. A consistency problem is called k^+-*restricted* if it is only defined for input samples that contain exactly k elements labeled with $+1$.

Given a negative sample S and an element $x \in U$, the *detector sampling distance* [9] $\Delta_{\mathcal{D}}(S, x)$ is defined by

$$\Delta_{\mathcal{D}}(S, x) = \begin{cases} \frac{|\mathcal{D}[S \cup \{(x,+1)\}]|}{|\mathcal{D}[S]|} & \mathcal{D}[S] \neq \emptyset \\ \bot & \text{otherwise} \end{cases}, \tag{1}$$

with $\bot$ denoting the undefined value. We previously showed the following.

Theorem 1 (Simulating NSAs via the Sampling Distance [9]). *If $\Delta_{\mathcal{D}}$ can be computed in expected polynomial time*[1]*, then there exists a randomized algorithm that is input-output-equivalent to* NegativeSelection(S, M, n) *and runs in expected polynomial time.*

Instead of simulating the NSA explicitly, we can also simply output $\Delta_{\mathcal{D}}(S, m)$ for every input element $m \in M$ and use this fraction as an "anomaly score". For this application, we will be satisfied if we can compute only the numerator of Equation 1, because the denominator is anyway equal for all $m \in M$. However, counting solution sets of combinatorial problems is often infeasible – we provided some according negative results previously [9]. Therefore, the method that we put forward in this paper rests on a slightly weaker precondition.

Proposition 1. *Suppose there exists an algorithm that, for every negative input sample $S \subseteq \mathcal{U} \times \{-1\}$, outputs a detector $\pi \in \mathcal{D}[S]$ sampled u.a.r. in expected polynomial time. Then there exists an input-output-equivalent algorithm for* NegativeSelection(S, M, n) *that runs in expected polynomial time in the size of S and M as well as in n.*

[1] Throughout the paper, this means that the expected runtime must be polynomial for *all* possible inputs.

Hence, by sampling directly from $\mathcal{D}[S]$, we can overcome the efficiency problems of rejection sampling. However, for achieving acceptable detection rates (in AIS) or for simulating realistic immune systems (in computational immunology), n often has to be very large. Therefore, we would like to avoid generating detectors explicitly. This too can be achieved if we sample from both $\mathcal{D}[S]$ *and* $\mathcal{D}[S \cup \{(x, +1\}]$, i.e., from the set of consistent detectors for both 0^+-restricted and 1^+restricted input samples.

Proposition 2. *Suppose there exists an algorithm as defined in Proposition 1 and another algorithm that, for every input sample $S \subseteq \mathcal{U} \times \{-1\}$ and every element $x \in \mathcal{U}$, samples a detector $d \in \mathcal{D}[S \cup \{(m, +1)\}]$ u.a.r. Furthermore, assume that the 0^+- and 1^+-restricted consistency problems for $\mathcal{D}$ are both "self-reducible" [15]. Then there exists a fully polynomial time randomized approximation scheme (FPRAS) for computing the detector sampling distance $\Delta_{\mathcal{D}}(S, m)$.*

Proof. For self-reducible problems, Jerrum et al. [15] showed that a polynomial time u.a.r. sampler[2] can be used to construct an algorithm that determines the number of solutions within factor $1 + \epsilon$ in polynomial time in both the input size and $1/\epsilon$. Applying this theorem, we obtain FPRASs for computing both the numerator and the denominator of Equation 1. Therefore, we can approximate $\Delta_{\mathcal{D}}$ within factor $1 + \epsilon'$, where $\epsilon' = 2\epsilon + \epsilon^2$. □

For lack of space, we cannot reproduce the precise definition of self-reducibility here, and refer the reader instead to Jerrum et al [15]. Intuitively, self-reducibility means that solutions to the whole problem can be constructed by extending solutions of slightly smaller instances. Self-reducibility seems to be "the rule rather than the exception" [15] for combinatorial problems. For instance, it is easy to show that all detector types that have so far been used in string-based negative selection (summarized in [9]) lead to self-reducible consistency problems.

Hence, via efficient detector sampling, we can approximate the results of detector compression techniques [8]. A natural question is therefore whether efficient detector sampling can be possible when efficient detector compression (which leads to detector counting algorithms) is not. In the upcoming section, we prove that this can indeed be the case.

Theorem 2. *There exists a detector type $\mathcal{D}$ for which (1) computing $\Delta_{\mathcal{D}}$ is #P-hard, and (2) there exist expected polynomial time algorithms for sampling u.a.r. from both $\mathcal{D}[S \cup \{(x, +1)\}]$ and $\mathcal{D}[S]$, implying an FPRAS for $\Delta_{\mathcal{D}}$.*

The classical example for a combinatorial problem where counting the solutions is #P-hard but sampling from the solution space is easy is DNF-satisfiability [15]. However, consistency problems correspond to conjunctions of constraints. Therefore, there appears to be no way to encode DNF-satisfiability in a 0^+- and 1^+-restricted consistency problem (without exponential blowup), which is the main technical challenge that needs to be overcome to prove Theorem 2.

[2] In fact, it is only required to sample approximately u.a.r. from $\mathcal{D}[S]$ and $\mathcal{D}[S \cup \{(x, +1)\}]$. However, in this paper we restrict ourselves to u.a.r. sampling.

4 Proof of the Main Theorem

We first have to prove a technical lemma.

Lemma 1 (Embedding an Additional Object into a Uniform Sampler). *Let X be a finite set of unknown cardinality $|X| > 1$, and suppose that there exists an algorithm $\mathcal{A}$ that generates an element of X u.a.r. in expected polynomial time. Let $x^* \notin X$. Then there exists an algorithm $\mathcal{A}^*$ that generates an element of $X \cup \{x^*\}$ u.a.r. in expected polynomial time.*

Proof. Let n denote the unknown cardinality of X. Our procedure $\mathcal{A}^*$ works as follows: (1) $\mathcal{A}^*$ samples an element $a \in X$ u.a.r. (2) $\mathcal{A}^*$ repeatedly samples a tuple $(x, y) \in X^2$ u.a.r. until $(x, y) \neq (a, a)$. (3) If $x = a$, then $\mathcal{A}^*$ outputs x^*. Otherwise, $\mathcal{A}^*$ outputs a. Now the probability that $\mathcal{A}^*$ outputs x^* is

$$\frac{n-1}{n^2-1} = \frac{1}{n+1} ,$$

and thus the output probability distribution is uniform over $X \cup \{x^*\}$. The lemma now follows by noting that because $|X| \geq 2$, step (2) above terminates after a constant expected number of iterations. □

Now we are prepared to prove Theorem 2.

Proof (Theorem 2). The basic idea is to define a detector type $\mathcal{D}$ whose consistency problem amounts to finding graph colorings. We recall that a k-coloring of a graph $G = (V, E)$ is a mapping $C : V \to [1, k]$, and it is called *valid* if $C(v) \neq C(w)$ for all $\{v, w\} \in E$. Counting the k-colorings of a graph with maximal degree κ is #P-hard for all constants $k, \kappa \geq 3$ [16]. Still, for $k > \kappa(\kappa + 2)$ there exists an algorithm that samples u.a.r. in expected polynomial time from the valid k-colorings of a given graph. Below, we assume that $\kappa \leq 3$ and $k \geq 16$. This includes the #P-hard case $\kappa = 3, k = 16$ as a special case, which will suffice to establish our hardness result. Note that a graph of maximum degree 3 is always 16-colorable, such that the decision version of the graph coloring problem is trivial for these constants. In the following, we denote the maximum degree of a given graph G by $\kappa(G)$.

To make the proof more palatable, we proceed in three steps. First we show that there exists a detector type $\mathcal{D}$ for which determining $|\mathcal{D}[S]|$ is #P-hard, even though we can sample u.a.r. from $\mathcal{D}[S]$ for 0^+-restricted samples S. This is not yet exactly what we need because $|\mathcal{D}[S]|$ is only the denominator of the detector sampling distance $\Delta_{\mathcal{D}}$ (see Equation 1); infeasibility of computing the denominator of a fraction does not imply infeasibility of computing the entire fraction. In the second step, we deal with this technicality. In the third step, we show the feasibility of sampling consistent detectors for both 0^+- and 1^+-restricted samples u.a.r.

Step 1. Our universe $\mathcal{U}$ is the set of graphs with maximum degree ≤ 3 and one labeled edge, which we call the *root edge* ρ:

$$\mathcal{U} = \{(V, E, \rho) : V = [1, L],\ E \subseteq \{e \subseteq V : |e| = 2\},\ \rho \in E,\ \kappa(V, E) \leq 3\} .$$

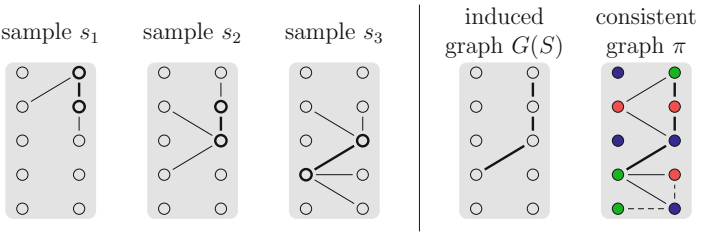

Fig. 2. Illustration of the proof of Theorem 2. The definition of $\mathcal{M}$ ensures that every edge $\{u, v\}$ sharing a node with a root edge in one sample must also occur in every other sample whose root edge contains u or v, otherwise there exists no S-consistent pattern. For example, if any of the non-root edges in sample s_2 were missing, then no pattern could be consistent with $(s_1, -1), (s_2, -1)$ and $(s_3, -1)$. The induced graph $G(S)$ is the union of all root edges. The pattern set $\mathcal{P}$ is the set of all colored graphs, and a graph $\pi \in \mathcal{P}$ is S-consistent if and only if it contains all edges from the samples and its nodes are validly colored with respect to $G(S)$. Edges not occurring in the samples may only occur in π if they do not touch any root edge from S, like the dashed edges in the rightmost graph.

Figure 2 shows three examples s_1, s_2, and s_3 (with roots edges ρ in bold).

We are going to define a detector type $\textsc{gcol} = (\mathcal{P}, \mathcal{M})$ whose pattern set $\mathcal{P}$ and matching function $\mathcal{M}$ will be constructed in such a way that a negative sample $S \subseteq \mathcal{U} \times \{-1\}$ encodes an *induced graph* $G(S)$. Counting S-consistent patterns will be equivalent to counting the valid k-colorings of $G(S)$. We define

$$\mathcal{P} = \{(V, E, C) : V = [1, L],\ E \subseteq \{e \subseteq V : |e| = 2\},\ C : V \to [1, k]\}\ ,$$

which is simply the set of all arbitrarily k-colored graphs with L vertices (the coloring need not be valid). The matching function is defined as follows:

$$\mathcal{M}_{\textsc{gcol}}((V, E, C), (V, E', \rho)) = \begin{cases} +1 & (1)\ E' \not\subseteq E \\ & \text{or } (2)\ E \setminus E' \text{ contains a } \rho\text{-adjacent edge} \\ & \text{or } (3)\ C \text{ is no valid coloring for } \rho \\ -1 & \text{otherwise} \end{cases}$$

Now consider a negative sample S, and suppose there does not exist an S-consistent pattern $\pi = (V, E, C)$. This can occur if and only if there exist two samples $(V, E_1, \rho_1), (V, E_2, \rho_2) \in S$ and an edge $e \in E_1$ such that e shares a node with ρ_1 but not with ρ_2. In other words, an S-consistent pattern exists if and only if it holds true that once an edge $\{u, v\}$ appears in one sample graph where either u or v belongs to the root edge, it appears in *every* sample graph where u or v belongs to the root edge. Therefore, the existence of an S-consistent pattern can be decided in polynomial time.

Consider a negative sample $S = \{((V, E_1, \rho_1), -1), \ldots, ((V, E_n, \rho_n), -1))\}$ for which at least one consistent graph pattern $\pi = (V, E, C)$ exists. We define the *induced graph* $G(S)$ as the union of all root edges in S, i.e., $G(S) := (V, \cup_i \{\rho_i\})$.

The definition of $\mathcal{M}$ directly ensures that (1) every S-consistent graph $\pi \in \mathcal{P}$ contains $G(S)$ as a subgraph, and (2) C is a valid coloring of $G(S)$. Moreover, it is not hard to show that $G(S)$ has maximum degree ≤ 3 – if that weren't the case, then there would either be no S-consistent pattern or one sample with maximum degree > 3. Edges that did not occur in S and do not share nodes with $G(S)$ may or may not be present in π (dashed edges in the consistent graph π in Figure 2), and nodes that are not in $G(S)$ (like the top left and bottom left nodes in the consistent graph in Figure 2) are assigned an arbitrary color.

Let $\#\chi(G(S))$ be the number of valid k-colorings of $G(S)$. Let ϵ denote the number of edges not adjacent to nodes in $G(S)$ (those which may or may not be present in S-consistent graphs). Then, assuming $|\textsc{gcol}[S]| > 0$, we have

$$|\textsc{gcol}[S]| = \#\chi(G(S)) \cdot 2^{\epsilon} \ .$$

Now suppose we had an algorithm for computing $|\textsc{gcol}(S)|$. Then we could determine the number $\#\chi(G)$ of valid k-colorings for an arbitrary graph G of maximal degree $\kappa = 3$ as follows: Decompose $G = (V, E)$ into sample graphs by creating for each edge $e = \{u, v\} \subseteq V$ a sample graph containing root edge e and its adjacent edges in G. Create a sample S containing all these graphs with negative labels. Then $G(S) = G$, and

$$\#\chi(G) = \frac{|\textsc{gcol}[S]|}{2^{\epsilon}}$$

gives the number of valid k-colorings of G. Because ϵ can be computed in polynomial time from S, computing $|\textsc{gcol}[S]|$ must thus be #P-hard[3] for $k \geq 3$.

Conversely, given an arbitrary negative sample S over $\mathcal{U}$, we can sample from $\textsc{gcol}[S]$ as follows. First check whether $|\textsc{gcol}[S]| = 0$ as discussed above. If this is not the case, we compute the induced graph $G(S) = (V, E)$, and sample a valid coloring of $G(S)$ u.a.r. using Huber's algorithm [14]. Next, consider every edge $\{u, v\} \subseteq V$ that does not occur in S and does not share nodes with root edges from S, and insert $\{u, v\}$ into E with probability $1/2$. The resulting graph is sampled u.a.r. from $\textsc{gcol}[S]$. An example result of this process is depicted as the rightmost graph in Figure 2.

Step 2. So far we proved that computing $|\textsc{gcol}[S]|$ is #P-hard. To show that computing $\Delta_{\textsc{gcol}}$ is also #P-hard, we insert a special pattern $\hat{\pi}$ into $\mathcal{P}$ and a special element $\hat{x}$ into $\mathcal{U}$ such that $\hat{\pi}$ matches only $\hat{x}$ and vice versa. From now on, let $\textsc{gcol}$ denote this augmented pattern class. Suppose we had access to an oracle that computes $\Delta_{\textsc{gcol}}(S, \hat{x})$. We could use this oracle to count the k-colorings of a graph $G = (V, E), V = [1, L]$, as follows: We create a negative sample S with $G(S) = G$. Then

$$\Delta_{\textsc{gcol}}(S, \hat{x}) = \frac{1}{1 + \#\chi(G)\, 2^{\epsilon}} \ .$$

[3] Strictly speaking, only functions to the natural numbers can be #P-hard. Therefore, more formally correctly we should say that every #P-hard function could be computed by a polytime algorithm with single-call access to an oracle for $|\textsc{gcol}[S]|$.

Hence, we could compute $\#\chi(G)$ from $\Delta_{\text{GCOL}}(S, \hat{x})$ by rearranging the above, which implies that computing $\Delta_{\text{GCOL}}(S, \hat{x})$ is #P-hard.

Step 3. It remains to show that for all $S \subseteq \mathcal{U} \times \{-1\}$ and for all $m \in \mathcal{U}$, we can generate elements of both $\text{GCOL}[S]$ and $\text{GCOL}[S \cup \{(m, +1)\}]$ u.a.r. in expected polynomial time. We start with the case where there is no positive sample. If S contains $\hat{x}$, then $\hat{\pi}$ is not S-consistent, and we output a pattern sampled at uniform from $\text{GCOL}[S]$. Otherwise, $\hat{\pi}$ is S-consistent, and we apply Lemma 1 to sample a pattern at uniform from $\text{GCOL}[S] \cup \{\hat{\pi}\}$. Now, consider the case where there is one positive sample $(m, +1)$. If m is a subgraph of the union of all sample graphs and the root of m occurs as a root in S, then there is no S-consistent pattern that matches m. Otherwise[4], m contains at least one edge e not present in S. We iteratively generate S-consistent patterns π u.a.r until we find one that matches m. Each π will match m with probability $\geq 1/2$, because the probability that π contains e is $1/2$. Therefore, after a constant expected number of trials we find the desired π, which is u.a.r. from $\text{GCOL}[S \cup \{(m, +1)\}]$.

5 Outlook

In this paper we presented a novel generic approach for implementing negative selection algorithms (NSAs) efficiently and proved that there exist cases where the new approach is feasible even though the detector compression technique that we put forward previously [9] is not. An obviously desirable next step would be to demonstrate the feasibility of this approach in practice. Sampling-based approximation algorithms, in particular Markov Chain Monte Carlo (MCMC) methods, have proved successful in many areas, e.g. Bayesian network analysis, even in cases where rigorous proofs for convergence in polynomial time (which can be notoriously difficult) are lacking. One appealing feature of MCMC approaches is their typical ease of implementation. This presents an advantage over detector compression, which often relies on rather intricate data structures [8]. To illustrate this, we conclude by defining an MCMC method for a natural detector type, Boolean monomials, and leave proving or disproving its efficient convergence as an open problem.

Open Problem. For $\mathcal{U} = \{0, 1\}^L$, $\mathcal{P} = \{0, 1, *\}^L$, let π match x if π and x are identical at all positions i where $\pi_i \neq *$ (π can be interpreted as a Boolean monomial). Given $S \subseteq \mathcal{U}^L \times \{-1\}$ and $m \in \mathcal{U}^L$, generate an $S \cup \{(m, +1)\}$-consistent pattern π as follows. Initialize $\pi = m$. Then, in each step, do nothing with probability $1/2$ and else, pick a position $i \in [1, L]$ u.a.r. If $\pi_i = *$, set $\pi_i = m_i$. Otherwise, replace π_i by $*$ and check whether the resulting pattern matches any $s \in S$. If yes, undo the change, else continue. This algorithm describes an ergodic Markov chain M whose stationary distribution is uniform over all $S \cup \{(m, +1)\}$-consistent patterns. Prove or disprove: M is rapidly mixing.

[4] For lack of space, we omit the required, but technical special treatment of $|V| < 5$.

References

1. Forrest, S., Perelson, A.S., Allen, L., Cherukuri, R.: Self-nonself discrimination in a computer. In: Proceedings of the IEEE Symposium on Research in Security and Privacy, pp. 202–212. IEEE Computer Society Press (1994)
2. 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 the National Academy of Sciences of the United States of America 90(5), 1691–1695 (1993)
3. Timmis, J., Hone, A., Stibor, T., Clark, E.: Theoretical advances in artificial immune systems. Theoretical Computer Science 403, 11–32 (2008)
4. Stibor, T.: Foundations of r-contiguous matching in negative selection for anomaly detection. Natural Computing 8, 613–641 (2009)
5. Stibor, T.: On the Appropriateness of Negative Selection for Anomaly Detection and Network Intrusion Detection. PhD thesis, Technische Universität Darmstadt (2006)
6. Aickelin, U.: Special issue on artificial immune systems: editorial. Evolutionary Intelligence 1(2), 83–84 (2008)
7. Elberfeld, M., Textor, J.: Efficient Algorithms for String-Based Negative Selection. In: Andrews, P.S., Timmis, J., Owens, N.D.L., Aickelin, U., Hart, E., Hone, A., Tyrrell, A.M. (eds.) ICARIS 2009. LNCS, vol. 5666, pp. 109–121. Springer, Heidelberg (2009)
8. Elberfeld, M., Textor, J.: Negative selection algorithms on strings with efficient training and linear-time classification. Theoretical Computer Science 412, 534–542 (2011)
9. Liśkiewicz, M., Textor, J.: Negative selection algorithms without generating detectors. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2010), pp. 1047–1054. ACM (2010)
10. Chao, D.L., Davenport, M.P., Forrest, S., Perelson, A.S.: A stochastic model of cytotoxic T cell responses. Journal of Theoretical Biology 228, 227–240 (2004)
11. Chao, D.L., Davenport, M.P., Forrest, S., Perelson., A.S.: The effects of thymic selection on the range of T cell cross-reactivity. European Journal of Immunology 35, 3452–3459 (2005)
12. Košmrlj, A., Jha, A.K., Huseby, E.S., Kardar, M., Chakraborty, A.K.: How the thymus designs antigen-specific and self-tolerant T cell receptor sequences. Proceedings of the National Academy of Sciences of the USA 105(43), 16671–16676 (2008)
13. Košmrlj, A., Read, E.L., Qi, Y., Allen, T.M., Altfeld, M., Deeks, S.G., Pereyra, F., Carrington, M., Walker, B.D., Chakraborty, A.K.: Effects of thymic selection of the T-cell repertoire on HLA class I–associated control of HIV infection. Nature 465, 350–354 (2010)
14. Huber, M.: Exact sampling and approximate counting techniques. In: Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing, STOC 1998, pp. 31–40. ACM, New York (1998)
15. Jerrum, M.R., Valiant, L.G., Vazirani, V.V.: Random generation of combinatorial structures from a uniform distribution. Theoretical Computer Science 43, 169–188 (1986)
16. Bubley, R., Dyer, M., Greenhill, C., Jerrum, M.: On approximately counting colourings of small degree graphs. SIAM Journal on Computing 29, 387–400 (1998)