International Journal of Engineering and Technical Research (IJETR)
ISSN: 2321-0869 (O) 2454-4698 (P), Volume-5, Issue-4, August 2016

A Compact Detector Set for Artificial Immune Systems

Nguyen Van Truong

***Abstract*— Negative selection algorithms (NSA) are methods inspired by the T cells maturating process. They all comprise of two phases: generation of detectors that match none of the self samples, and classification of monitored elements as self or nonself using these detectors. However, the detector sets generated may be redundant. In this paper, we propose a new negative selection algorithm to generate a complete and non-redundant detector sets that use an extension of r-chunk matching rule. This allows to reduces detectors storage and classification time. Experimental results on four datasets show the effective of proposed algorithm.**

***Index Terms*—Immune system, negative selection, r-chunk, detectors generation.**

I. Introduction

In the field of Artificial Immune Systems (AIS), negative selection algorithm is class of techniques inspired by the T cells maturating process that happens in thymus. The discriminating mechanism between self (signal of a healthy cell) and nonself (signal of an unhealthy cell) of T cells are modeled by NSAs. T cells are first generated randomly and in a large number, in the hope that every pathogen that might infect the host could be detected by at least some of these cells. However, the host must ensure that no cell generated would turn against itself (autoimmune reactions). Hence, newborn T cells must undergo the process of selection to ensure that they are able to recognize nonself. This process might be conducted by a negative selection: if a T cell detects any self protein, it is discarded; otherwise, it survives [4].

Given a collection of self patterns S, a typical NSA comprises of two phases: detector generation and detection [2], [12]. In the detector generation phase (Fig. 1.a), the detector candidates are generated randomly and censored by matching them against given self samples taken from the set S. The candidates that match any element of S are eliminated and the rest are kept and stored in the set D. In the detection phase (Fig. 1.b), the collection of detectors are used to distinguish self (system components) from nonself (outlier like viruses, worms, etc.). If an incoming data instance matches any detector, it is claimed as nonself, and it is claimed as self otherwise.

From a machine learning perspective, negative selection is usually described as an anomaly detection technique. Since its introduction, NSA has been a source of inspiration for many computing applications, especially for intrusion detection [4], [14], computer virus detection [9], monitoring UNIX processes [8], spam detection [18], modeling of immunological processes such as HIV infection modeling [15].

Nguyen Van Truong, Faculty of Mathematics, Thai Nguyen University of Education, Thai Nguyen City, Vietnam, Mobile No. +(84)-0915016063.

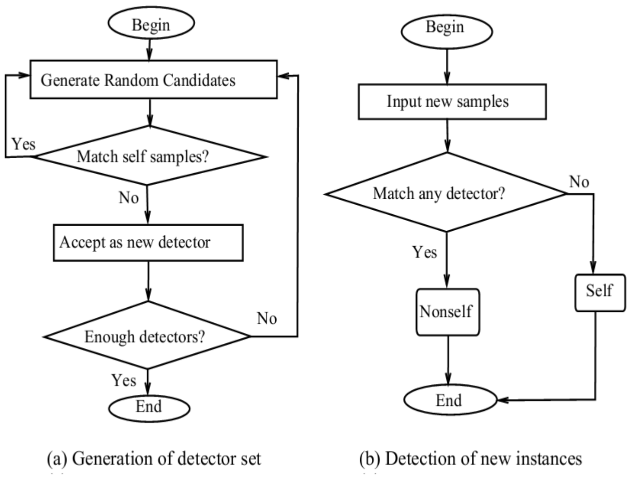

Fig. 1. Outline of a typical negative selection algorithm [13].

For binary-based AIS (i.e. the representations for cells, detectors are binary strings), r-chunk and r-contiguous bits (rcb) are two common matching rules used for the construction of detector set (and also for the detection phase). An r-chunk matching rule can be seen as a generalization of the rcb matching rule. To date, there have been some computing models of binary detectors that could generate a complete and non-redundant detector set, called perfect detector set, a set of minimum detectors with the same detection ability in comparing to that of all possible detectors, such as those based on prefix trees [5], [16], or automata [6]. In these models, detectors are represented as a whole structure (tree or automata) rather than a set of individual strings. While they provide a more compact representation of the detector sets for AIS and therefore achieve a better detection time complexity, these models of binary-based AIS are hard to deploy in distributed environments. Naturally, one desirable property of NSA is its ability to be implemented in a distributed manner - each detector might detect different kind of nonself, this is desirable for many applications such as in computer security systems. Therefore, the focus of this paper is on binary-based NSAs that employ a discrete set of detectors (strings) so that they can be implemented in distributed environments. We can, for example, randomly divide the discrete set of detectors into some subsets, each one for a nodes in in distributed environments.

With respect to binary-based AIS using discrete detector set, to the best of our knowledge, the only algorithm for generating a perfect and discrete set of r-chunk (rcb-based) detectors was proposed by T. Stibor in [21] (by S. T. Wierzchoń in ([20]), which has frequently been cited, compared, and applied in the literature with 44 (47) citations on Google Scholar. The main contribution of our paper is to design new deterministic algorithm to generate a perfect and discrete set of rcbvl-based detectors, which is equitable to a full set of r-chunk-based detectors in term of anomaly detection.

Moreover, compact string-based detectors set can achieve better memory and time complexities compared to conventional algorithms.

The rest of the paper is organized as follows. In the next section, we first present some basic terms and definitions. After the introduction of strings, two common matching rules for generating detector set, r-chunk and rcb are given. Then we introduce an r-contiguous bit matching rule with variable length (rcbvl) detectors. This new type of detector set is more compact, in bits, than one bases on original r-chunk and rcb. Prefix trees are introduced as temporary data structures for generation. Section 3 details our new NSA that can generate perfect detector sets base on rcbvl. Section 4 briefly describes our experiments in generating perfect detector sets. Section 5 concludes the paper and discuss some possible future works..

II. Basic Terms And Definitions

In NSAs, an essential component is the matching rule which determines the similarity between detectors and self samples (in the detector generation phase) and coming data instances (in the detection phase). Obviously, the matching rule is dependent on detector representation. In this paper, both self and nonself cells are represented as binary strings of fixed length. This representation is the most simple and popular representation for detectors and data in AIS, and other representations (such as real valued) could be reduced to binary [10], [13].

A. Strings

An alphabet Σ is nonempty and finite set of symbols. A string $s \in \Sigma^*$ is a sequence of symbols from Σ, and its length is denoted by |s|. A string is called empty string if its length equals 0. Given an index $i \in \{1,...,|s|\}$, then s[i] is the symbol at position i in s. Given two indices i and j, whenever $j \geq i$, then s[i...j] is the substring of s with length j− i+1 that starts at position i and if j < i, then s[i...j] is the empty string. A string s' is a prefix of s if s' = s[1...j], $1 \leq j \leq |s|$.

Given a string $s \in \Sigma^{\ell}$, a non-empty string d, and an index $i \in \{1,..., \ell - r + 1\}$, we say that d occurs in s at position i if s[i...i + |d| − 1] = d. Moreover, concatenation of two strings s and s' is s + s'.

Although our approaches can be implemented on any finite alphabet, but strings used in all examples are binary, Σ = {0,1}, just for easy understanding.

B. R-chunk and rcb matching rules

For binary-based AIS, the rcb and r-chunk are among the most common matching rules. Given a positive integer r, a set S of self strings of length ℓ. A detector under rcb matching rule is a string of length ℓ that does not match any $s \in S$. It is said to match another string, of the same length, if they have r consecutive matching bits in the corresponding positions. Rcb was introduced and used in many AIS projects [7], [11], [17]. An r-chunk detector is a tuple of a string of r bits and its starting position with the string that does not match any $s \in S$. An r-chunk detector (d,i) is said to match a string s if d is a prefix of s[i..|s|]. An r-chunk matching rule is considered as a simplification of the rcb matching rule [17]. This type of detector helps AIS to achieve better results on data where adjacent regions of the input data sequence are not necessarily semantically correlated, such as in network data packets [3]. It is noted that an r-contiguous detector [4] can be decomposed into ℓ − r + 1 overlapping r-chunk detectors.

Example 1: Let ℓ = 6, r = 3. Given a set of five self strings S = {s_1 = 010101, s_2 = 111010, s_3 = 101101, s_4 = 100011, s_5 = 010111}. The set of all r-chunk detectors is {(000,1), (001,1), (011,1), (110,1), (001,2), (010,2), (100,2), (111,2), (000,3), (100,3), (111,3), (000,4), (001,4), (100,4), (110,4)}. The set of all detectors under rcb matching rule is {001000, 001001, 011110, 110000, 110001}.

C. Rcbvl matching rule

Given a positive integer r, a set S of self strings of length ℓ. A triple (d,i,j) of a string $d \in \Sigma^k$, $1 \leq k \leq \ell$, an integer $i \in \{1,...,\ell-r+1\}$ and an integer $j \in \{i,...,\ell-r+1\}$ is called a negative detector under rcbvl matching rule if d does not occur in any s, $s \in S$. In another words, (d,i,j) is an rcbvl detector if there exist j−i+1 r-chunk detectors (d_1 ,i),..., (d_{j-i+1}, j) that d_k , d_{k+1} are two (r − 1)-bit overlapping strings, k = 1,...,j − i.

Example 2: Given ℓ, r and the set S of self strings as in Example 1. Triple (0001,1,2) is an rcbvl detector because there exist two 3-chunk detectors (000,1), (001,2) that 000 and 001 are two 2-bit overlapping strings. A perfect detector set D under rcbvl matching rule contains 5 variable length detectors {(0001,1,2), (00100,1,4), (100,4,4), (011110,1,4), (11000,1,3)}. It is a minimum detectors set (23 bits) that covers all detector space of r-chunk detectors set in Example 1 (45 bits).

D. Prefix trees

A prefix tree T is a rooted directed tree with edge labels from Σ where for all σ ∈ Σ, every node has at most one outgoing edge labeled with σ. For a string s, we write s ∈ T if there is a path from the root of T to a leaf such that s is the concatenation of the labels on this path. This tree structure is important for generating rcbvl detectors set.

Example 3: Given ℓ, r and the set of self strings S as in Example 1, four prefix trees presenting all binary 3-chunk detectors are in Fig. 2.

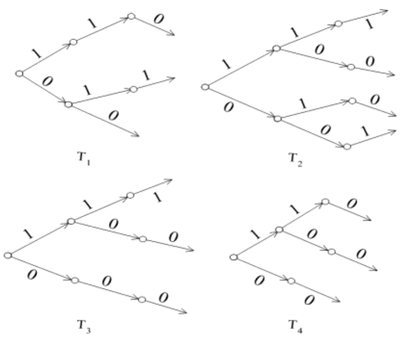

Fig.2. Trees represents 3-chunk detectors set in Example 1. Tree T_i presents 3-chunk detectors (d,i), i = 1,...,4.

III. New Negative Selection Algorithm

Given a non-empty set S of self strings of length ℓ, and an integer $r \in \{1,...,\ell - r + 1\}$, this section presents a new NSA bases on rcbvl matching rule.

A. Detectors set generation under rcbvl matching rule

Algorithm 1 Algorithm to generate perfect rcbvl detector set.

```
1: procedure G ENERATION D ETECTORS (S, ℓ, r, D)
2: for i = 1,...,ℓ − r + 1 do
3:     Create an empty prefix tree $T_i$
```

```
4: end for
5: for all $s \in S$ do
6:    for i = 1,...,$\ell - r + 1$ do
7:       insert every $s[i...i + r - 1]$ into $T_i$
8:    end for
9: end for
10: for i = 1,...,$\ell - r + 1$ do
11:   for all nonleaf node $n \in T_i$ and all $\sigma \in \Sigma$ do
12:      if no edge with label $\sigma$ starts at n then
13:         create a new leaf n' and an edge (n,n')
            labeled with $\sigma$.
14:      end if
15:   end for
16:   delete every node $n \in T_i$ from which none of the
      newly created leaves is reachable.
17: end for
18: $D_1 = \emptyset$
19: $D = \{(s,1,1)|s \in T_1\}$
20: for i = 2,...,$\ell - r + 1$ do
21: $D_2 = \emptyset$
22: for all $(s,k,j) \in D$ do
23:   if there exists a $s' \in T_i$ where $s[i-k+1...|s|]$
      is prefix of it then
24: $D_2 = D_2 \cup \{(s+s'[|s|-j+k...|s'|],k,i)\}$
25: delete every node $n \in T_i$ from which only
     nodes in the s' is reachable
26: for all $s' \in T_i$ where $s[i - k + 1...|s|]$ is
     prefix of it do
27:   if $|s| - i + k < r$ then
28:      $D_2 = D_2 \cup \{(s[|s|] + s', i - 1, i)\}$
29:   else
30:      $D_2 = D_2 \cup \{(s', i, i)\}$
31:   end if
32: delete every node $n \in T_i$ from which
     only nodes in the s' is reachable
33: end for
34: else
35:   $D_1 = D_1 \cup \{(s,k,j)\}$
36: end if
37: end for
38:   for all $s' \in T_i$ do
39:      $D_2 = D_2 \cup \{(s', i, i)\}$
40:   end for
41: $D = D_2$
42: end for
43: $D = D \cup D_1$
44: end procedure
```

Algorithm 1 summarizes the first phase of new NSA. Some prefix trees are first used to generate perfect detectors set from S and then this set is used to distinguish if a new sample as self or nonself. In the algorithm, the process of generating We first construct for every position $i \in \{1,...,\ell - r + 1\}$ a prefix tree T_i . Each prefix tree T_i can be constructed as follows: start with an empty prefix tree and insert every $s[i...i + r - 1]$, $s \in S$, into it (lines 5-9). Next, for every non-leaf node n and every $\sigma \in \Sigma$ where no edge with label σ starts at n, create a new leaf n' and an edge (n,n') labeled with σ. Finally, delete every node from which none of the newly created leaves is reachable (lines 10-17). Detectors set D is first created by all s $\in$ T 1 in line 19. The rest of the algorithm, lines 20-42, updates partial detectors in D by identifying their right overlapping strings in prefix trees.

From the description of the algorithm, it takes $|S|.(\ell-r+1).r$ steps to generate $(\ell-r+1)$ prefix trees and $|D|.(\ell-r+1).2^r$ steps to generate perfect detector set D.

Example 4: Given ℓ, r and the set of self strings S as in Example 1. Some steps in the Algorithm 1 generating a perfect detector set as in Example 2 are: Set D is first created as (00,1,1), (011,1,1), (110,1,1). Then the for loop (lines 20-42) calculates D and D_1 as following:

For i = 2: D = (0001,1,2); (0010,1,2); (0111,1,2); (1100,1,2) and $D_1 = \emptyset$. For i = 3: D = (00100,1,3); (01111,1,3); (11000,1,3) and D_1 = (0001,1,2). For i = 4: D = (00100,1,4); (011110,1,4) and D_1 = (0001,1,2); (11000,1,3); (100,4,4). The final step, $D = D \cup D_1$ in line 43, generates the perfect detector set {(0001,1,2), (00100,1,4), (100,4,4), (011110,1,4), (11000,1,3)}.

B. Detection under Rcbvl matching rule

To detect if a given string s is self or nonself, we simply check our Rcbvl matching rule on s against every detector in D. If it is the case, output s is nonself, otherwise s is self. The function min used in Algorithm 2 to return the smallest number from two values. It is easy to see that this algorithm has the same time complexity with Algorithm 1.

Algorithm 2 Algorithm to detect if a given string s is self or nonself.

```
1: procedure DETECTION (s, $\ell$, r, D)
2: for all $(s',n,m) \in D$ do
3:  for i = n,...,m do
4:     if $s'[i...\min(i + r - 1,|s'|)]$ occurs in s at position i then
5:          output s is nonself
6:          exit procedure
7:       end if
8:    end for
9: end for
10: output s is self
11: end procedure
```

IV. Experiments

We use a popular flow-based datasets NetFlow [19] and a random dataset for experiments. The flow-based NetFlow is generated from packet-based DARPA dataset [1] is used for experiment 1. This dataset focuses only on flows to a specific port and a IP address which receives the most number of attacks. It contains all 129,571 traffics (including attacks) to and from victims. Each flow in the datasets has 10 fields: Source IP, Dest. IP, Source Port, Dest. Port, Packets, Octets, Start Time, End Time, Flags, and Proto. Similar to the previous studies [19], we select the same 4 features Packets, Octets, Duration and Flags from the NetFlow dataset as the input of two experiments in case study 1. A randomly created dataset is used for experiments in case study 2. This dataset contains 50,000 binary string with the length of 30.

Flows in NetFlow are converted into binary strings by two steps. The first step is to map all features to binary string features. After this step, a total string features are constructed for both normal data and anomalous one. The second step is to concatenate the binary string features for every flows. After this step, dataset contains binary strings with their length of 49. The distributions of training and testing datasets as well as parameters r, ℓ for 4 experiments are described in Table 1.

Table1. Data and parameters distribution for experiments and result comparison.

ℓ	r	Train	Test	Size (bit)		Time (mil. Sec.)	
				r-chunk	rcbvl	r-chunk	rcbvl
Case 1							
49	10	119571	10000	206810	42704	58962	11960
49	8	79571	50000	31672	8096	48952	11085
Case 2							
30	12	25000	25000	367092	79222	243979	44995
30	14	40000	10000	2324056	392815	518999	82922

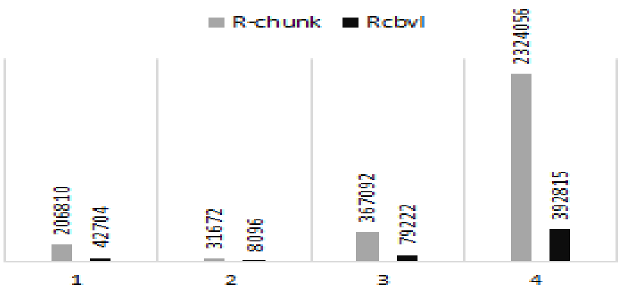

Fig. 3. Size of detectors comparisons

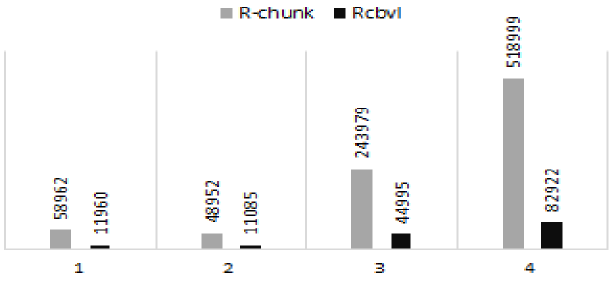

Fig. 4. Classification time comparisons

Results in Table 1 show that our proposed algorithm reduce both size (bits) of detectors and time (milliseconds) to classify testing dataset. The comparisons of detection time and detectors' size are illustrated in Fig. 3 and Fig. 4, respectively.

V. Conclusion

In this paper, we have proposed a novel NSA to generate perfect detector sets for string-based AIS. We developed a rcbvl matching rule as an extension of traditional rcb. Our new algorithm has a polynomial time complexity. More importantly, proposed algorithm always generate complete and non-redundant detector sets for string-based AIS. Experiment results show that proposed algorithm can reduce both detection phase time complexity and storage of detectors. Moreover, the varying length of the parameter r in the rcbvl matching rule can balance specialization and generalization in classification systems, which will be the next step of our study. How to apply the algorithm to intrusion detection systems would be our interesting research direction.

Acknowledgment

Research reported in this publication was supported by the ASEAN-European Academic University Network (ASEA-UNINET).

References

[1] DARPA dataset. http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/index.html

[2] A. S. A. Aziz, M. Salama, A. ella Hassanien, and S. E. O. Harafi. Detectors generation using genetic algorithm for a negative selection inspired anomaly network intrusion detection system. In Proceedings of the FedCSIS'2012, pp. 597–602, 2012.

[3] J. Balthrop, F. Esponda, S. Forrest, and M. Glickman. Coverage and generalization in an artificial immune system. In Genetic and Evolutionary Computation Conference (GECCO), pp. 3–10, 2002.

[4] D. Dasgupta. Artificial Immune Systems and Their Applications. Springer-Verlag, Berlin Heidelberg, 1998.

[5] M. Elberfeld and J. Textor. Efficient algorithms for string-based negative selection. In International Conference on Artificial Immune Systems, pp. 109–121, 2009.

[6] M. Elberfeld and J. Textor. Negative selection algorithms on strings with efficient training and linear-time classification. Theoretical Computer Science, 412(6):534 – 542, 2011.

[7] F. Esponda, S. Forrest, and P. Helman. The crossover closure and partial match detection. In International Conference on Artificial Immune Systems (ICARIS), pp. 249–260. Springer-Verlag, 2003.

[8] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff. A sense of self for UNIX processes. In IEEE Symposium on Research in Security and Privacy, pp. 120–128, 1996.

[9] S. Forrest, A. S. Perelson, L. Allen, and R. Cherukuri. Self-nonself discrimination in a computer. In IEEE Symposium on Security and Privacy, pp. 202–212, 1994.

[10] F. González, D. Dasgupta, and J. Gómez. The effect of binary matching rules in negative selection. In Genetic and Evolutionary Computation Conference (GECCO), pp. 195–206, 2003.

[11] S. Hofmeyr. An immunological model of distributed detection and its application to computer security. PhD thesis, The University of New Mexico, ALbuquerque, NM, 1999.

[12] Z. Ji. Negative Selection Algorithms: from the Thymus to V-detector. PhD thesis, The University of Memphis, August 2006.

[13] Z. Ji and D. Dasgupta. Revisiting negative selection algorithms. Evolutionary Computation, 15:223–251, 2007.

[14] J. Kim, P. J. Bentley, U. Aickelin, J. Greensmith, G. Tedesco, and J. Twycross. Immune system approaches to intrusion detection - a Review. Natural Computing, 6:413–466, Dec. 2007.

[15] A. Komrlj, E. Read, Y. Qi, T. Allen, M. Altfeld, S. Deeks, F. Pereyra, M. Carrington, B. Walker, and A. Chakraborty. Effects of thymic selection of the T-cell repertoire on HLA class I-associated control of HIV infection. Nature, 465:350–354, 2010.

[16] V. T. Nguyen, X. H. Nguyen, and C. M. Luong. A novel combination of negative and positive selection in artificial immune systems. In IEEE International Conference on Computing and Communication Technologies, Research, Innovation, and Vision for the Future (RIVF), pp. 6–11, 2013.

[17] J. K. Percus, O. E. Percus, and A. S. Perelson. Predicting the size of the T-cell receptor and antibody combining region from consideration of efficient self-nonself discrimination. volume 90, pp. 1691–1695, 1993.

[18] Y. Tan. Anti-Spam Techniques Based on Artificial Immune System. CRC Press, 2016.

[19] Q. A. Tran, F. Jiang, and J. Hu. A real-time netFlow-based intrusion detection system with improved BBNN and high-frequency field programmable gate arrays. In IEEE International Conference on Trust, Security and Privacy in Computing and Communications, pp. 201–208, 2012.

[20] S. T. Wierzchoń. Generating optimal repertoire of antibody strings in an artificial immune system. In IIS'2000 Symposium on Intelligent Information Systems, pp. 119–133, 2000.

[21] T. Stibor, K. M. Bayarou, and C. Eckert, "An investigation of R-chunk detector generation on higher alphabets," in Genetic and Evolutionary Computation Conference (GECCO), vol. 3102 of Lecture Notes in Computer Science, pp. 299–307, 2004.

Biography

Nguyen Van Truong is a lecturer in the Faculty of Mathematics at Thai Nguyen University of Education, from where he received a Bachelor of Mathematics and Informatics in 2000. He finished his master course on Computer science at Vietnamese National University in 2003. He is currently a PhD student at Institute of Information Technology (IOIT), Vietnamese Academy of Science and Technology (VAST). He has taught a wide variety of courses for UG students and guided several projects. He has published several papers in National Journals & International Conferences. His research interests are embedded systems and artificial immune systems.