Immune Anomaly Detection Enhanced with Evolutionary Paradigms

Marek Ostaszewski
Faculty of Sciences, Technology and Communication
University of Luxembourg
6 rue Coudenhove Kalergi
L-1359
Luxembourg-Kirchberg, Luxembourg
marekostaszewski@o2.pl

Franciszek Seredynski
Polish-Japanese Institute of Information Technology
Koszykowa 86
02-008 Warsaw, Poland
Institute of Computer Science, Polish Academy of Sciences
Ordona 21
01-237 Warsaw, Poland
sered@ipipan.waw.pl

Pascal Bouvry
Faculty of Sciences, Technology and Communication
University of Luxembourg
6 rue Coudenhove Kalergi
L-1359
Luxembourg-Kirchberg, Luxembourg
pascal.bouvry@uni.lu

ABSTRACT

The paper presents an approach based on principles of immune systems to the anomaly detection problem. Flexibility and efficiency of the anomaly detection system are achieved by building a model of network behavior based on the self-nonself space paradigm. Covering both self and nonself spaces by hyperrectangular structures is proposed. Structures corresponding to self-space are built using a training set from this space. Hyperrectangular detectors covering nonself space are created using niching genetic algorithm. A coevolutionary algorithm is proposed to enhance this process. Results of experiments show a high quality of intrusion detection, which outperform the quality of recently proposed approach based on hypersphere representation of self-space.

Categories and Subject Descriptors: I.2.8 [Problem Solving, Control Methods, and Search]: Heuristic methods

General Terms: Algorithms, Experimentation.

Keywords: Artificial Immune Systems, Coevolution, Network Anomaly Detection.

1. INTRODUCTION

An artificial immune system (AIS) is a computational paradigm based on abstracting natural immunological processes [6], which can be applied to solve problems of computer security, including detection of intrusions and anomalies [5,7,14]. Classical approach bases on the recognition of attack signatures [11]. AIS offers alternative mechanisms to deal with unwanted activities in computer networks, including generalization and recognition of previously unknown attacks. A recent promising approach to network anomaly detection has been presented in [3], and is based on a description of legitimate behavior using hypersphere structures. We propose to use hyperrectangle structures to provide a more precise definition of the normal traffic and coevolutionary-based mechanisms to enhance the process of anomaly detection.

The paper is organized as follows: the coming section contains a short definition of AIS and its associated mechanisms. The principles of construction of an effective model of network behavior, called self space, are presented in section 3. Genetic algorithm used to generate detectors of nonself space, along with niching and coevolutionary techniques are described in section 4. Section 5 contains results of experiments, which were carried out using sets of network data collected at MIT. Last section contains conclusions and a discussion of further possibilities of development of the presented approach.

2. IMMUNE ANOMALY DETECTION

Proper classification system is required in order to assess the network behavior, and the Self-Nonself paradigm offered by AIS methodology seems to be an accurate choice. The self space corresponds to the organism protected by its natural immune system, which consequently cannot attack or define as enemy any defended cells. In network domain, the self space is defined on basis of normal traffic and the nonself space contains all possible threats and deviations. They are complementary, the same way it is for an organism and its environment in nature. Therefore, the definition of normal traffic is based on an existing subset of regular data, and incoming traffic that does not match these definitions will be considered as abnormal. Self-Nonself space principles are coupled with a negative selection algorithm [5], which is used to construct detectors focused on nonself space. They are used later during the monitoring process to capture abnormal patterns in network traffic. An accurate model of space is required to effectively deal with anomaly detection. Authors in [13] indicate that the Hamming shape-space suffers from several weaknesses. In effect the model of space presented in [3] has been taken under consideration. Although authors in [12] indicate that real-valued negative selection may cause severe problems for system scalability and detector set generation, a similar approach presented in this paper seems to be worth of taking under consideration.

3. PROPOSED SELF SPACE CONSTRUCTION

Self space, henceforth called *Self* is constructed using values of monitored parameters from legitimate traffic, and some principles of defining it has been presented in [3]. Because of potentially having different ranges, every value is normalized to the $[0.0, 1.0]$ interval, and n parameters give n-dimensional space. Every combination of them, defined as a vector of values $\vec{x} = (x^1, \dots, x^n)$ will be called a state of a system, belonging to the space of all possible states of the system, $[0.0, 1.0]^n$, henceforth called *S*. The subspace complementary to *Self* containing all possible threats and anomalies will be defined as $Nonself = S - Self$.

The self space construction is based on recorded regular traffic and every of its states becomes automatically an element of *Self*. To implement threshold affinity a definition of the variability parameter (henceforth called $\boldsymbol{v}$) is needed. An assumption that certain states are similar to recorded normal states allows to define a level of deviation from given data.

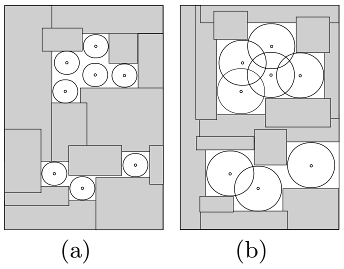


Figure 1: Self space and detector sets for (a) low and (b) high value of $\boldsymbol{v}$

In [3], the authors proposed to use a spherical shape for *Self* characterized by $\boldsymbol{v}$ being the radius of a hypersphere centered in the recorded state. A spherical construction of *Self* has a drawback when it comes to the detector construction: covering *Nonself* with hyperspheres may not be effective, as shown in [12]. Moreover, considering the detector construction proposed below (i.e. using hyperrectangular structures), the spherical *Self* is inappropriate because the generated detectors have no chance to cover the Nonself space completely. Figure 1 illustrates this case. As one may see, subspaces are created that cannot be covered by the detectors due to shape conflict between the structures describing *Self* and *Nonself*.

Figure 2 presents our approach to construct a hyperdimensional space, where a dimension (in this case 2D) depends on a number of parameters taken into consideration. $\boldsymbol{V}$ describes the similarity degree, therefore *Self* can be defined as a subspace that contains all states with some level of resemblance to recorded normal states. Hyperspherical construction of *Self* cannot distinguish between the different dimensions providing only one value (hyperradius) for every dimension. Hyperrectangular construction provides different values of $\boldsymbol{v}$ components for each dimension and allows constructing more accurate *Self*. Henceforth, $\boldsymbol{v}$ is defined as a vector $\boldsymbol{v} = (\boldsymbol{v}^1, \dots, \boldsymbol{v}^n)$. For a given state of the system $\vec{x} = (x^1, \dots, x^n)$ and $\boldsymbol{v}$ the structure of self space takes the form of two vectors: $low = (x^1 - \boldsymbol{v}^1, \dots, x^n - \boldsymbol{v}^n)$ and $high = (x^1 + \boldsymbol{v}^1, \dots, x^n + \boldsymbol{v}^n)$, where a pair low^j and $high^j$ describes an interval for the parameter j considered as normal. A state of the system fitting in all intervals belongs to the normal space.

Figure 2(a) illustrates *Self* for a relatively small value of $\boldsymbol{v}$, which implies that the detectors of *Nonself* have a greater possibility of raising a false alarm (false positive error), but less risk of leaving any anomaly undetected (false negative error) [4]. On the other hand, Figure 2(b), shows a higher false negative ratio, with lower number of false positives. Figure 2(c) presents a *Self* constructed for two combined $\boldsymbol{v}$ values, and shows, that the detector sets are different, like in the *Nonself* classification process. One can see that $\boldsymbol{v}$ used for construction of *Self* influences the tolerance, and allows to construct complex security model.

Multiple detector sets that correspond to the different levels of security are applied by assessing every state of the system with all generated sets and defining, which one raises an alarm [3]. The security levels have to be sorted out by growing value of $\boldsymbol{v}$ that were used to construct *Self*. The highest level defines the most tolerant set to abnormalities. Using more than one set, the *Nonself* classification process is changed and the range of returned values is widened using $classify(\vec{x}) = max(\{level(DSets)\} \cup \{0\})$, where $level(DSets)$ returns the highest index of detector sets that raises an alarm. The system states used for the

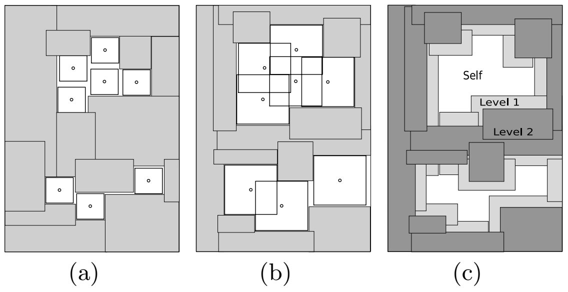


Figure 2: Self space and detector sets for (a) low and (b) high value of $\boldsymbol{v}$, and (c) detection levels in summary

construction of *Self* are recorded during a defined period and create time series. The anomaly detection process is highly dependant on the frequency of recording and on the processing monitored values. Therefore, in order to improve the precision of the *Self* construction process, the sliding window method is applied. Time series of recorded values $R = r_1, r_2, \dots, r_n$ for monitored parameters with a certain frequency are transformed to window series $W = \{(r_1, \dots, r_w), (r_2, \dots, r_{w+1}), \dots, (r_{n-w+1}, \dots, r_n)\}$, and may also take the form $W = \{rw_1,\ rw_2,\ \dots, (rw_m)\}$, where $m = n - w + 1$ and $rw_i = r_i/w\ + r_{i+1}/w\ + \cdots + r_{i+w}/w$ with w as a parameter describing a window size. Applying sliding window method allows to define the notation of the system state as $\vec{x} = (rw_i^1,\ rw_i^2, \dots,\ rw_i^n)$. Authors in [3] apply different kind of sliding window fashion to achieve aggregations between parameters. A number of search space dimensions is increased by treating succeeding values of monitored parameters in sliding window as additional dimensions. Therefore monitoring n parameters with window size w creates $n * w$-dimensional state space and greatly increases computational complexity.

The values of the parameters are measured every period and during this process two subsets of states are built, training set and testing set. The parameter $\boldsymbol{v}$ is calculated from the standard deviation based on the distances between states belonging to training set, and the testing set is used to adjust *Self*. It is achieved by checking, if created structures intercept every state of the testing set. If it is not the case, a new structure based on the uncaught state is created, and the process is restarted from the beginning.

4. DETECTOR SET GENERATION

In our scheme, anomaly detection is a process of monitoring *Nonself* by using detectors having a hyperrectangular structure. *Self* elements are created from recorded states with $\boldsymbol{v}$ and their number is potentially large in the case when a long period is used to model *Self*. Therefore, it is desirable to develop a set of structures (detectors) efficiently covering *Nonself*, with a potentially large volume and a small overall number. The detector set is described by [3] $DSet = \{D_1, \ldots, D_k\}$, where D_i : *if Condition then Nonself*, and $Condition = x^1 \in [low_i^1, high_i^1] \wedge \ldots \wedge x^n \in [low_i^n, high_i^n]$. A single detector classifies if a given state is captured by the covered space defined in the conditional part by two vectors, *high* and *low*. The construction of a space from junction of intervals in every dimension is similar to the case of *Self* elements (see Section 3).

4.1 Niching genetic algorithm

Creating a detector set is a complex multiobjective problem. In order to provide good solutions [3] proposes an approach based on genetic algorithm (GA). The goal is to cover the space as large as possible with detectors that avoid *Self* elements and cover unique space, without overlaying the same *Nonself*. A population of GA consists of individuals constructed from a pair of vectors, reflecting *high* and *low* values of a conditional part of a detector. Therefore, real-coded GA [9] have to be applied to this problem, as long as vectors of real values (from $[0.0, 1.0]$ interval) are used. GA schema goes as follows [3]:

```
DetectorSet ds=$null$; numAtt=0; numDet=0;
while numDet < maxDet and numAtt < maxAtt
    $runGA(W, \boldsymbol{v})$;
    $D \leftarrow$ best evolved detector;
    $fit = calculateFitness(D)$;
    if $fit >$ minFit
        $ds.addDetector(D)$;
        $numAtt = 0$;
    else
        $numAtt = numAtt + 1$;
end while
return ds,
```

where W is a learning set consisting of *Self* states, $\boldsymbol{v}$ is a variability parameter used for *Self* structures construction, `minFit` is the minimal value of fitness expected from evolved detector, `maxAtt` and `numAtt` are the maximal and the current number of attempts to evolve a single detector, respectively and `maxDet` and `numDet` are the maximal and the current number of detectors in the detector set, respectively.

GA has to deal with multiobjective problem and for this reason niching GA (NGA) [1] has been applied to cover different subspaces of *Nonself* with volumes as large as possible. During each run of NGA the fitness function is modified in order to focus the process of searching for new detectors in subspaces not covered by the previously evolved detectors. NGA goes as follows:

```
SolutionSet solSet = null;
while not solSet.$satisfied()$
Population = $runGA()$;
for each Individual $\rightarrow$ ind from Population
    $Individual.calculateFitness()$;
    for each $Individual \rightarrow sol$ from $solSet$
        $sim = calculateSimilarity(ind, sol)$;
        $Individual.decreaseFitness(sim)$;
    if $ind.fitness() >$ minFitness
    $solSet.add(ind)$;
end while
return solSet.
```

The fitness is computed by taking under consideration three factors, as shown below:

- **Volume calculation** - a volume of a given detector is calculated as follows:

$$Volume(D) = \prod_{i=1}^{n} (high^i - low^i),$$

where *high* and *low* are elements of the vectors on the positions corresponding to the used parameters.

- **Overlaying with *Self* structures** - a volume of the space overlayed with *Self* is computed as follows:

$$SelfOverlay(D) = \sum_{i=1}^{m} (D \cap xs_i),$$

where xs_i is a structure created from a state coming from the learning set W and variability parameter $\boldsymbol{v}$, as described in Section 3.

- **Overlaying with already developed detectors** - a volume of the space overlayed by already developed detectors is calculated as follows:

$$DetectorOverlay(D) = \sum_{i=1}^{k} (D \cap D_k).$$

The fitness of a single detector is calculated from the equation

$$Fitness(D) = Volume(D) - (SelfOverlay(D) + DetectorOverlay(D)) .$$

4.2 Coevolutionary mechanism in the detector generation process

Although NGA tends to cover all *Nonself* in the most efficient way, this process is unsupervised and the goal for created detectors is based only on constraints. The criterion of volume does not specify, where exactly in search space detector should lay, therefore a goal is not precisely defined. Additional information could lower the computational cost of the generation process and, in effect, give more specialized detectors covering certain subspaces in *Nonself*. For that purpose a coevolutionary algorithm is used. Coevolution is relatively new research area in the field of evolutionary computation. The basic idea is taken from the world of Nature, where two or more coexisting species are constraining one other to evolve better features. Among many coevolution models, one seems to be useful to the detector generation problem. *Predator - prey* paradigm [10] describes a model, where individuals of one type (predators) are trying to catch individuals of another type (preys). The population of the first species develops features that allow it to catch its prey easily, and attributes of the second one evolve to make escape from a predator possible. Applying this process to the detector generation mechanism could improve it, by providing a certain goal in search space. Coevolution allows controlling the process, by enforcing on generated detectors certain features, indicating areas to cover. Some definitions [10] have to be assumed to apply coevolutionary algorithm to detector generation problem:

- **Constraint Satisfaction Problem (CSP):** a class of problems effectively solvable by coevolutionary algorithms. The first of two coevolving populations is a population of solutions (henceforth called *Solutions*), and the other one is a population of constraints (henceforth called *Constraints*) that *Solutions* have to fit. Because of their static nature, *Constraints* cannot evolve but their fitness can be also evaluated.
- **Encounter:** a confrontation between individuals from *Solutions* and *Constraints* results in the a victory of one and the loss of the other. A *Solution* wins if it fits given *Constraint*, and loses if *Constraint* cannot be satisfied by a given solution.
- **LifeTime Fitness Evaluation (LTFE):** in opposite to the classic GA, every individual is tested multiple times and has a list of his encounters that changes, as it might be said, through its lifetime. A fitness is calculated on the basis of confrontations with individuals from coevolving population. LTFE regulates a number of confrontations, and thus, affects calculated fitness. Probability of choice to encounter depends on fitness, therefore, even if *Constraints* cannot evolve, winning ones are tested more frequently against *Solutions*.

To apply the coevolutionary algorithm to the detector generation problem, the second (coevolving) population has to be assumed. To define proper constraints for detectors, a set of anomalies has to be constructed. Similarly to the *predator - prey*, the proposed model considers a situation when individuals from the detector set try to intercept individuals from a set of anomalies. An anomaly (individual) is defined as a certain state from *Nonself* space in the form of a vector $a = (a_1, ..., a_n)$, where a_i is a value for corresponding parameter. An encounter between a detector and an anomaly leads to an assessment checking if the anomaly is placed inside the subspace covered by the detector. The detector wins the encounter if it intercepts the given anomaly, otherwise the constraining state is the winner.

A fitness of a given detector is finally computed after running the coevolutionary algorithm as follows:

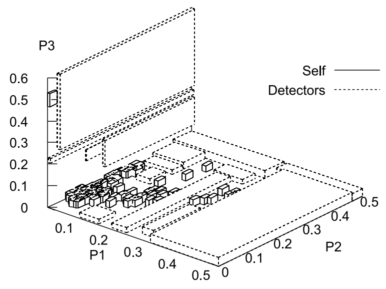
$$Fitness(D) = Fitness(D) + EncounterHistory(D) * Fitness(D) \ ,$$

where $EncounterHistory(D)$ is a function returning the summarized effect of all encounters for a given detector.

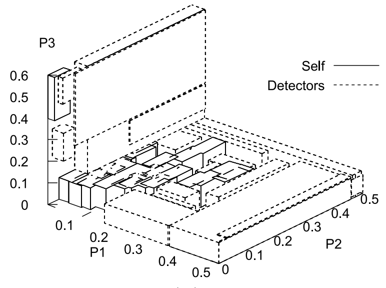
This way generated individuals are focused on certain subspaces, and specialized in capturing states from *Nonself* defined as the coevolving population. This process resembles vaccination, the way the specialization of antibodies in the human immune system is achieved by presenting negative examples.

5. EXPERIMENTAL RESULTS

A number of experiments have been performed to find out the effectiveness of nonself approach to the anomaly detection problem, based on hyperrectangular *Self* structures and involving the coevolutionary algorithm. The *Self* space for this experiment was constructed using data gathered at MIT [8]. The first week of the collected network traffic using *tcpdump* was unaffected by anomalies, and for one of chosen computers figuring IP 172.16.114.50 (marx), the states of network parameters were collected and used for the *Self* structures development. The network traffic parameters used for experiments are the number of bytes per second (P1), the number of packets per second (P2) and the number of ICMP packets per second (P3), and a state of the system takes form of the vector $state = \{P1, P2, P3\}$ including values measured for the given IP address. The *Self* structures for GA were created from the first week data in a proportion 70:30 of training to testing sets.



(a)



(b)

Figure 3: Self space and detector sets for (a) 0.3*v* and (b) 1.0*v*

5.1 Experiment #1 - Generation of the detector set for different *v* values

In the first experiment GA was used for development of self space and detector sets for different ***v*** levels. NGA was run with max. num. of runs equal to 20, max. num. of attempts to evolve rule equal to 15, a number of generations equal to 750 and a population size equal to 100. The following GA operators have been used [9]: a tournament selection with the tournament size equal to 2, vector crossover, Gaussian mutation with probability 0.1 and border mutation with border values 0 and 1, and probability 0.01.

Figure 3 shows the comparison between the covered *Non-self* space with the detectors developed for the case of two values of $\boldsymbol{v}$. Figures 3(a) and 3(b) show only a part of the generated detectors. The total size of the detector set in both cases is equal to 15.

5.2 Experiment #2 - Anomaly detection process for different *v* and *w* values

Detector sets generated by NGA have been used to anomaly detection process on MIT data [8]. The second week contains five simulated attacks, one for every day of the network traffic, as shown in Table 1.

Table 1: MIT Second week attacks

Day	Name	Type	Start	Duration
1	Back	DoS	9:39:16	00:59
2	Portsweep	Probe	8:44:17	26:56
3	Satan	Probe	12:02:13	02:29
4	Portsweep	Probe	10:50:11	17:29
5	Neptune	DoS	11:20:15	4:00

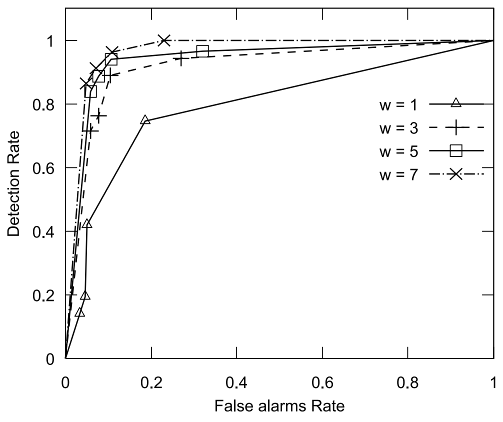


Figure 4: ROC diagram for different w sizes

A ROC (Receiver Operating Characteristics [4]) diagram presented on Figure 4 shows the classification performance of the detector sets for a given window size equal to 1, 3, 5 and 7, respectively. Points marked on each curve correspond to values of $\boldsymbol{v}$ equal to 1.3, 1.0, 0.7 and 0.3, respectively. One can notice that for given window size detection rate grows, when value of $\boldsymbol{v}$ decreases. It is worth noticing that the window size influences the precision of detection, and that the detector set constructed for w=7 performs well even for relatively large $\boldsymbol{v}$, what results in decreasing the number of false alarms.

The results of monitoring the anomaly detection process are presented in Figures 5 and 6. Figure 5(a) presents anomalies detected by the set of detectors for 0.3$\boldsymbol{v}$ and w=1, and Figure 5(b) for w=3. Figure 6 presents detection effects for the set with 1.0$\boldsymbol{v}$ and w=1 (Figure 6(a)), and w=3 (Figure 6(b)). The analysis of this Figure indicates greater sensitivity of detectors constructed for Self with w=3, what can be explained, if temporal patterns are taken under consideration. With larger window size, one can intercept time dependencies between preceding and succeeding states, what is impossible for detectors based on w=1.

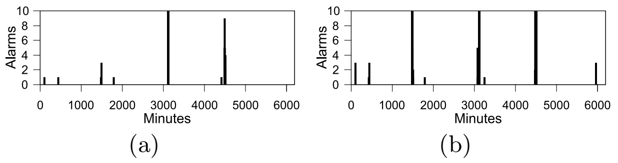


Figure 5: Attacks detected using 0.3$\boldsymbol{v}$ and w equal (a) one and (b) three

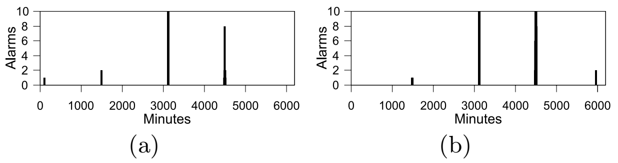


Figure 6: Attacks detected using 1.0$\boldsymbol{v}$ and w equal to (a) one and (b) three

Figure 6 indicates that combining both detector sets results in discovering all five attacks, though having relatively high level of $\boldsymbol{v}$. As one may notice, some peaks in these Figures (1485 and 4491 minutes in Figure 5(a), 4498 in Figure 6(a) and 1487 in Figure 6(b)) are groups of multiple lines. After the analysis of Table 1 it is possible to notice, that the duration of attacks 2 and 4 was relatively long and the system raised more than one alarm during the monitoring process. Those attacks on Figures 5(b) and 6(b) are indicated as groups of lines having alarm number equal to or more than 10, what makes them look like bold lines. An interesting fact can be observed after the comparison of alarms raised for each attack - obviously, probe attacks are recognized with greater accuracy than DoS attacks. Additionally, the window size seems to have optimal values for every attacks, as it may be observed in Figure 6, where w=1 manages to capture the first attack, but misses the last one, and for w=3 the first attack remains unreported, but last one is displayed. Interesting case is the attack number three, indicated with a great strength in every parameter configuration though relatively short duration time. It may be explained by dependencies between an attack type and the structure and parameters used in *Self* construction.

5.3 Experiment #3 - Anomaly detection process for spherical construction of *Self*

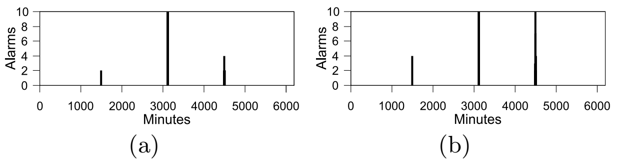


Figure 7: Sphere detectors efficiency for 0.5$\boldsymbol{v}$ and w equals (a) one and (b) three

The approach presented in [3] is based on detector sets developed for *Self* constructed of hyperspheres, which are created using a given state of the system as a center and single value of $\boldsymbol{v}$ as a hyperradius. Experimental results presented in [3] show, that a hyperspherical design of *Self* was sufficient to catch four attacks at most, with a window size equal to 3, and three with a window size equal to 1. It is worth emphasizing that due to different sliding window construction in [3], w=3 defined nine dimensional space, making detector generation computationally more expansive. Experiments carried out for this paper include also the construction of *Self* with hyperspheres, and the detector generation for this purpose. Figure 7 presents detected anomalies for $0.5\boldsymbol{v}$ with w=1 (Figure 7(a)) and w=3 (Figure 7(b)). Because of the proposed way of computing $\boldsymbol{v}$, the results differ from those presented in [3], while in both cases with hyperspherical the *Self* construction system was unable to discover all five attacks. The comparison with the detector sets based on hyperrectangular structure indicates that the approach presented in this paper is more precise and offers better performance for anomaly detection.

5.4 Experiment #4 - Coevolution effectiveness for randomly generated set of anomalies

Mechanisms of coevolution has been tested to check, if there is a possibility of applying it to enhance the detector generation process. A set of 1000 randomly generated vectors (Set A) from *Nonself* has been assumed as the second population coevolving with the population of generated detectors. After generation of a detector set using both coevolutionary and classic NGA, this set has been tested against the Set A, to check how many of elements have been caught. The detector generation process has been slightly altered for coevolutionary NGA, including max. num. of attempts, or catching all anomalies as conditions of finishing a generation process.

Table 2: Performance of detector sets for randomly generated set of anomalies

Generations	LTFE	Anomalies caught
100	5	866
	10	899
	20	**928**
	—	895
300	5	981
	10	962
	20	**982**
	—	972
500	5	963
	10	961
	20	973
	—	**987**

Table 2 presents the results of the performed experiments, and the best of them have been highlighted by using bold fonts. The detector sets have been developed for three different number of generations and for three different values of LTFE parameter. One additional detector set has been generated using standard NGA, without coevolution mechanism, marked with "—" symbol in LTFE column. The number of anomalies that have been caught differs, but the gain brought by coevolutionary NGA is insignificant, and for 500 generations classic NGA surpasses one with coevolution. These results can be explained by the random generation of anomaly set. The distribution of states in *Nonself* is regular, and classic NGA, while trying to cover largest space possible, intercepts also states generated without any specialization.

5.5 Experiment #5 - Coevolution effectiveness for a specialized set of anomalies

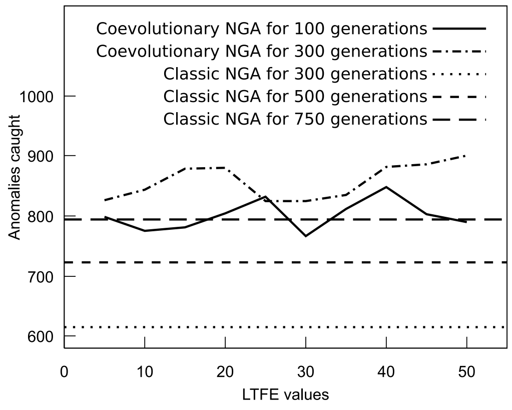


Figure 8: Comparison of coevolution efficiency with classic NGA approach

An alternative set of anomalies (Set B) has been generated for the detector set developed with the classical NGA, and used as coevolving population in the coevolutionary mechanism. Set B was specialized by generating its elements in *Nonself* space and beyond subspace covered by this given detector set. The distribution of anomalies from Set B is irregular and some of its elements belong to areas, where detectors are harder to develop, for example, in small subspaces between *Self* structures. Results of the conducted experiments are presented in Figure 8. The second experiment including coevolution shows a significant advantage of coevolutionary NGA and proves that the additional population can stimulate the detector generation process. Even relatively small number (100) of generations allowed the coevolutionary NGA to obtain better results than the classic NGA with 750 generations and, consequently with less computational cost. Furthermore, one can notice that the efficiency of coevolution is LTFE dependent, but also depends on number of generations, and results for more than 300 generations are worth further study. Another concern is a tradeoff between coevolution efficiency and detector fitness, calculated on the basis of various factors (see Section 4), which may cause worse performance of coevolutionary detectors development.

5.6 Experiment #6 - Coevolution effectiveness for less restrictive detector overlay criteria

The influence of detector overlay factor on coevolution performance has been examined. The detector generation criteria were less strict, allowing overlying a certain percent of its volume between detectors. Set B was used for the coevolution mechanisms. Figure 9 illustrates dependencies between LTFE factor and the number of anomalies caught for three certain overlay tolerance levels. As one may see, efficiency of capturing anomalies rises with tolerance for other detectors in the set, what can be explained by the difference of goals between NGA and the coevolutionary mechanism.

The first one tends to cover a space as large as possible, without overlaying already covered detectors. For the second one the goal is to capture certain points enclosed in certain subspaces, and these are more important than the volume of detector. Therefore, the development of coevolutionary stimulated detectors that includes constraints of classical NGA, restrains them from covering anomalies, if it would lead to overlap detector spaces.

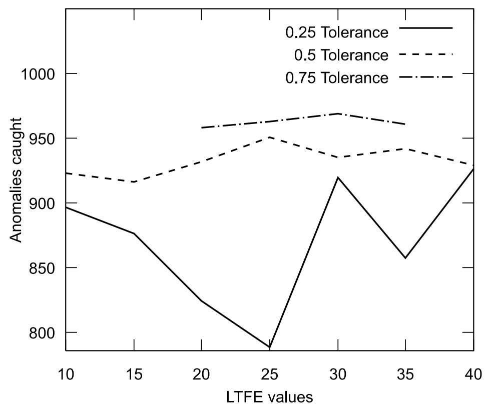


Figure 9: Comparison of coevolution efficiency for detector overlay tolerance

It is worth noticing that the growing level of overlay tolerance is related to the smoothness of lines, along with the efficiency of detector set. It proves that high sensitivity for overlapping of detectors may interfere and cause bad performance of coevolutionary stimulation in search for certain anomalies. Only three LTFE values are presented for tolerance level 0.75, showing the most significant peak for a given interval. These values seem to be optimal for the corresponding experiments presented below.

5.7 Experiment #7 - Initial population generation process

Results presented in experiment #6 suggest, that some of the anomalies are impossible to reach for detectors overlapping with each other, even for high tolerance level. This suggests, that unreachable points lay in vicinity of self structures, and are impossible to capture, which results from the criterion of avoiding self space. To improve performance of NGA looking for nonself detectors, a new initial population generation mechanism has been developed. For every run of NGA, a population of initial detectors was generated in such way, that none of them overlap with self space structure, or any already developed detector. This process (henceforth called constrained generation) has been applied to the coevolutionary NGA to search for new detectors rather in uncovered space, and to lower the probability of the development of improper detectors, overlapping with self, or detector structures. Figure 10 illustrates performance of coevolutionary NGA based upon a constrained initial population, and the results for all three LTFE values are very similar. Although constrained generation offers greater efficiency, not all Set B was covered. This indicates that some of anomalies are close enough to self states and still cause interferences for nonself detectors while intercepting them. Constraint generation seems to be a good method for narrowing the search space, but not sufficient. LTFE factor has small influence on interception process of the most difficult group, even taking into account a greatly increased fitness due to the captured anomalies. It plays no role if the detector covers *Self* space.

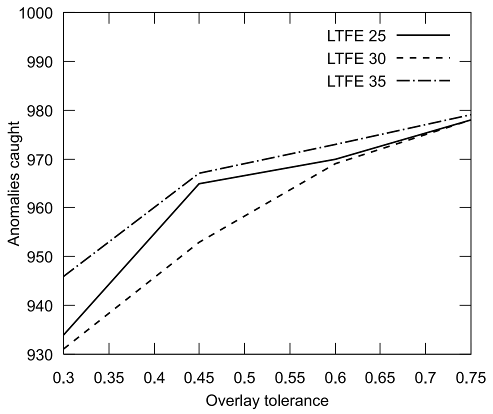


Figure 10: Coevolution efficiency for constrained generation of detectors

Recent work [2] suggests that gene libraries can be an efficient mechanism for improving coevolution process, therefore to provide population of NGA with sufficient information, a different method of initial population generation has been designed (henceforth called library generation). Only one library of information has been constructed, treating elements of Set B as information about the detector construction. Vertebrate immune systems develop antibody detectors using libraries of genes, providing immature antigens with certain knowledge and information. As mentioned in Section 3, detector consists of two vectors, *high* and *low*, and library generation process is based on a specific construction of detectors, assuming for those vectors two randomly chosen, but different anomaly points. Detectors constructed this way will not necessarily be correct, but infuse certain information into population. This kind of detector is constructed only with certain probability, which grows with number of detectors developed, due to succeeding reduction of anomalies set, and those most difficult that are left to find. Figure 11 presents the results of experiment involving library generation. Probability increment after single detector development was equal to 0.01.

As one may notice, the performance is better using a library-generated population, and for value of starting probability 0.3 and 0.35 it was possible to capture all presented anomalies by the developed detector sets. This mechanism seems to be efficient and using a properly constructed anomaly set it can boost the efficiency of entire detection system.

Figure 12 presents results of comparing three different methods of initial population generation: unsupervised, constrained and library generation. All methods seem to have similar progress, but constrained generation method offers better results, while library generation is the best in terms of number of covered anomalies. Analysis of Figure 12 indicates that group of about 980 anomalies is easier to reach by generated detectors, and they present groups, that can be covered by a single detector. The group of anomalies possible to reach only by library-generated population is spread into subspaces that are hard to reach: every new detector manages to cover only few of them, and over half of all non-

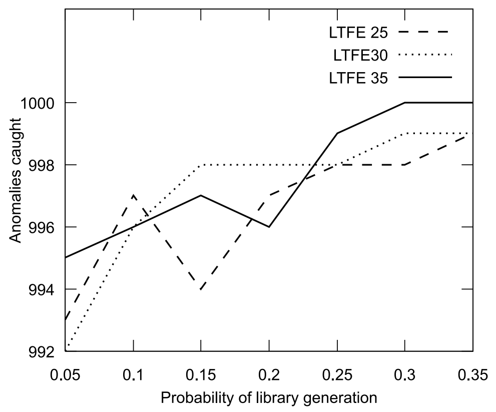


Figure 11: Coevolution efficiency for library generation of detectors

self detectors developed with last method is covering these 20 anomalies. It proves that coevolution assures both effectiveness and accuracy, because a low number of detectors capture anomalies easier to cover, and those that are difficult to cover are captured as well, but with larger number of detectors.

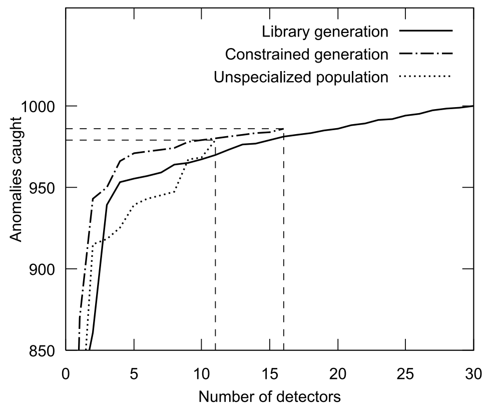


Figure 12: Comparison of efficiency of three different initial population generation processes

6. CONCLUSION

Results of conducted experiments indicate, that detectors generated with NGA proved to be effective, and hyperrectangular *Self* structures construction made a precise detection process possible. The presented approach is efficient and allows capturing all five kinds of simulated attacks in MIT data. While the hyperspherical design of *Self*, as presented in [3] made possible to catch only four out of five attacks, and applied for *Self* development and NGA presented in this paper, three of them. It also has been shown that the coevolutionary mechanisms can enhance the detectors generation process and in the result can make detection process more effective against given patterns of attack. Gathering data about some of those patterns in the form of coevolving sets can give in effect detector sets containing knowledge about attack subspaces. This mechanism can be compared to vaccine, which makes natural immune system more effective against certain illnesses.

Variability parameter $\boldsymbol{v}$ has been proven to be an important factor in the detector development process by influencing the *Self* volume. This parameter is responsible for adjusting the false alarms levels. Therefore, an algorithm of calculating $\boldsymbol{v}$ from learning is very important in the attempt to improve the detection ability of a system, and application of statistical approach presented in [12] seems to be interesting in this context.

Further research may involve different parameter types and greater number of them. The analysis of the detection process data shows, that the system performs very effective in the case of Satan attack in a relatively short duration time, and has more problems with attacks like Portsweep or Neptune, although their duration last several times longer (see Table 1). Looking for dependencies between parameters and attack types is also promising field of the research.

7. REFERENCES

[1] D. Beasley, D. R. Bull, and R. R. Martin. A sequential niche technique for multimodal function optimization. *Evolutionary Computation*, 2(1):101–125, 1993.

[2] S. Cayzer, J. Smith, J. Marshall, and T. Kovacs. What have gene libraries done for ais? In *Proceedings of the 4th International Conference on Artificial Immune Systems*, 2005.

[3] D. Dasgupta and F. González. An immunity-based technique to characterize intrusions in computer networks. *IEEE Transactions On Evolutionary Computation*, 6(3):1081–1088, 2002.

[4] T. Fawcett. Roc graphs: Notes and practical considerations for data mining researchers. *Technical Report HPL-2003-4*, 2003.

[5] S. Forrest, A. Perelson, L. Allen, and R. Cherukuri. Self-nonself discrimination in a computer. In *Proceedings of IEEE Symposium on Research in Security and Privacy*, 1994.

[6] S. M. Garret. How do we evaluate artificial immune systems? *Evolutionary Computation*, 13(2), 2005.

[7] M. Glickman, J. Balthrop, and S. Forrest. A machine learning evaluation of an artificial immune system. *Evolutionary Computation*, 13(2), 2005.

[8] http://www.ll.mit.edu/IST/ideval/index.html.

[9] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, 1992.

[10] J. Paredis. Constraint satisfaction with coevolution. In *New Ideas in Optimization*. McGraw-Hill, 1999.

[11] M. Roesch. Snort - lightweight intrusion detection for networks. In *Proceedings of the 13th Systems Administration Conference*, 1999.

[12] T. Stibor, J. Timmis, and C. Eckert. A comparative study of real-valued negative selection to statistical anomaly detection techniques. In *Proceedings of the 4th International Conference on Artificial Immune Systems*, 2005.

[13] T. Stibor, J. Timmis, and C. Eckert. On the appropriateness of negative selection defined over hamming shape-space as a network intrusion detection system. In *Proceedings of the 4th International Conference on Artificial Immune Systems*, 2005.

[14] S. T. Wierzchon. *Artificial immune systems. Theory and application (in polish)*. Exit, Warsaw, Poland, 2001.