A Solution Concept for Artificial Immune Networks: A Coevolutionary Perspective

Oscar Alonso, Fabio A. Gonzalez, Fernando Niño, and Juan Galeano

Intelligent Systems Research Lab - National University of Colombia
{omalonsom,fagonzalezo,lfninov,jcgaleanoh}@unal.edu.co

Abstract. In this paper a relation between artificial immune network algorithms and coevolutionary algorithms is established. Such relation shows that these kind of algorithms present several similarities, but also remarks features which are unique from artificial immune networks. The main contribution of this paper is to use such relation to apply a formalism from coevolutionary algorithms called *solution concept* to artificial immune networks. Preliminary experiments performed using the aiNet algorithm over three datasets showed that the proposed solution concept is useful to monitor algorithm progress and to devise stopping criteria.

1 Introduction

Several artificial immune network (AIN) algorithms have been developed based on Jerne's idiotypic network theory. They have been applied with success to several problems, including clustering, classification, optimization, and domain specific problems [3]. These algorithms present several similarities with evolutionary algorithms: an AIN is a population based meta-heuristic which uses variation and selection mechanisms over a population of individuals. Moreover, an AIN has elements in common with coevolutionary algorithms, a branch of evolutionary algorithms in which individuals interact between them. In an AIN, the stimulation of a cell is influenced not only by the antigens it detects, but also by other cells of the AIN.

Therefore, analyzing the relation between AINs and coevolutionary algorithms may provide interesting insights to each domain. Coevolution has been subject of theoretical analysis [4], which could provide AINs with effective stopping criteria, performance metrics, and a strong theoretical background. Also, coevolutionary algorithms may be improved by taking elements from AINs, such as diversity introduction (metadynamics), memory mechanisms, and dynamic regulation of population size. Some works have already successfully explored the relation between such models[10].

In this work AIN algorithms are analyzed from a coevolutionary point of view, by identifying common elements in both models, and also examining the differences between AIN algorithms and coevolutionary algorithms. Later, the *solution concept* formalism from coevolutionary algorithms is applied to AIN models, in order to characterize the expected result of the AIN dynamics. In addition, preliminary experimentation is performed on an existing AIN algorithm

L.N. de Castro, F.J. Von Zuben, and H. Knidel (Eds.): ICARIS 2007, LNCS 4628, pp. 35–46, 2007.

(aiNet), in order to test the potential use of the proposed solution concept to define stopping criteria and performance metrics.

The rest of the paper is organized as follows. Section 2 presents a background of coevolutionary algorithms. Section 3 analyzes artificial immune networks from a coevolutionary point of view, showing similarities and differences between such two models. In section 4, the solution concept formalism from coevolutionary algorithms is applied to AINs, and a solution concept is defined for AINs. Results of exploratory experimentation are shown in section 5, which suggests the potential use of the proposed solution concept to monitor the behavior of AIN algorithms . Finally, some conclusions are devised in section 6.

2 Coevolutionary Algorithms

An evolutionary process leads to an increase on the average fitness of a population. The main mechanisms of evolution are variation and selection; selection is performed according to a fitness measure, favoring better fit individuals, i.e., fittest individuals have a higher probability to survive.

Coevolution occurs when a change in individuals from one species induces evolution on individuals of another species. For example, suppose that a plant is subject to attack by an insect. The plant population may undergo evolution to develop toxins to defend itself from insects. Then, such toxins can induce evolution on the population of insects, for them to develop resistance to the toxin. Thus, the evolution of the plants induces a selection pressure over the insects population, which results in the evolution of the population of bugs.

From a computational viewpoint, a coevolutionary algorithm is characterized by a fitness function that changes as the population evolves, and is usually defined based on interaction with other individuals whom are also evolving.

2.1 One vs Many Populations

Coevolutionary algorithms usually involve more than one population of individuals. Each population represents a species, and the individuals of one population are evaluated according to their interaction with individuals of other populations.

It is also considered that coevolution arises when an individual is evaluated according to its interaction with individuals of its own population.

2.2 Competitive vs Cooperative Coevolution

The interaction between individuals can have different purposes, which can be mainly classified in two kinds of interaction: competitive and cooperative.

In competitive coevolution, each individual represents one solution to a problem, and the interaction between individuals is used to test individuals' features: the solution obtained is said to be test-based. An example of this interaction is when players for games are evolved: individuals receive fitness according to the results of their games against other individuals.

In cooperative coevolution, the output of the algorithm is composed of several individuals, which together represent a solution to the problem. Individuals interact in order to measure not only their features, but also their synergy. Here, the solution is said to be compositional. A very simple example of this approach is the optimization of a two variables function *f(x,y)*: the solution can be decomposed into two elements (x and y coordinates) and each value is evolved in its own population. Individuals are evaluated by selecting a random mate from the other population and evaluating the target function *f(x,y)*.

It is important to notice that in some cases the interaction presents characteristics from both categories: then, the interaction is said to be mixed.

2.3 Interaction Between Individuals

In most coevolutionary algorithms the interaction between individuals is direct: individuals meet each other and obtain fitness from such encounter. But, in some coevolutionary models, the interaction between individuals is not direct: their interaction occurs through their relation with a common environment, for instance, by consuming resources in a common environment. An example is the Lotka-Volterra model, in which predators interact directly with preys (by eating them), but they also interact by competing for a scarce resource: when a predator eats a pray, the remaining amount of prays for other predators diminishes.

Some well-known techniques in evolutionary algorithms which involve indirect interaction can be considered coevolutionary. An example is a niching technique known as *competitive fitness sharing*[4], used in multi-objective optimization. In this technique, the fitness produced by satisfying an objective is distributed among the individuals that are able to fulfill it. Thus, individuals that satisfy objectives that others do not are rewarded, promoting diversity in the population.

2.4 Generational vs Steady State Algorithms

In generational evolutionary algorithms, all individuals of the population are replaced by their descendants. In contrast, in steady state evolutionary algorithms, some individuals are added to the population and then some are removed, allowing an individual to remain in the population for several generations.

2.5 Spatially Embedded Algorithms

Usually, the interaction between individuals happens by choosing other individuals at random from their respective population. However, in some cases, individuals are embedded in a space, and the interaction only occurs between individuals located in a neighborhood. Frequently, such space corresponds to a lattice, and a local rule only allows individuals to interact with a limited amount of their neighbors [8].

The interaction space does not need to be explicit: it can be implicitly defined by a metric that represents a similarity measure between individuals. For instance, a niching technique known as *similarity based fitness sharing* [4] involves

interaction between individuals of the population, and, therefore, it can be considered coevolution. In this technique, the fitness of an individual decreases as the number of similar individuals in the population increases. It should be emphasized that in this case individuals interact according to locality in the genotype space.

In general, spatial embedding has been used to improve diversity in coevolutionary algorithms. Thus, spatiality does not represent any feature in the problem domain, but it is simply a convenient tool to improve an algorithm's performance[11].

2.6 Solution Concepts

A solution concept is a set of criteria used to determine which elements of a search space can be considered as solutions to a particular search problem. A solution concept partitions the search space into solution regions and non-solution regions. Therefore, a population-based algorithm should be designed in such a way that the population converges to solution regions. It is important to notice that a solution concept is specific to a particular search problem.

Every search problem defines a solution concept, which characterizes what is being searched. Also, each search algorithm implements a solution concept, which corresponds to what it actually searches. Sometimes, coevolution dynamics generates unexpected behavior. According to Ficici [4], this is related to a mismatch between the intended solution concept and the algorithm's solution concept. Therefore, solution concepts are a formalism useful in the design of proper algorithm dynamics, performance measures, and stopping criteria.

3 Immune Networks and Coevolution

Idiotypic network theory (also known as immune network theory) was developed by Jerne [6,9] in order to explain the memory and distributed self regulated behavior of the immune system. Such theory states that immune system cells are able to react not only to pathogens, but also to other elements of the immune system. During antigen exposure, clones (groups of identical cells) which recognize the antigen increase their size. When the antigen disappears, most of those cells would die and the clones would shrink or even disappear if not stimulated. Immune network theory states that a B cell can be stimulated by other cells of the immune system, and that this feedback mechanism can maintain the cells that were created during exposure to an antigen.

3.1 Artificial Immune Networks

An artificial immune network (AIN) is a computational model built upon Jerne's idiotypic network theory. It is a population based meta-heuristic, which develops a set of detectors (B cells[1]) that interact with data (antigens) and with each

[1] As each B cell express only one kind of antibody, some models consider antibodies directly, instead of B cells.

other. AINs perform unsupervised learning; they have been typically used for clustering, but have also been adapted to optimization, classification and domain specific applications [5].

AIN dynamics include interaction with antigens and between detectors. As a result of those interactions detectors become stimulated. Highly stimulated B cells undergo cloning and somatic hypermutation. Somatic hypermutation is a genetic operator in which the mutation rate depends on the affinity of the cell with the current stimulating antigen. This process is regulated by the interaction between B cells, which can stimulate them in order to create a memory of observed antigens, or suppress them, in order to control the immune response.

AIN algorithms also posses a *metadynamic*s, which includes natural death of unstimulated detectors and addition of new random detectors to the population.

3.2 A Comparison Between AINs and Coevolution

Here, similarities between AIN and coevolutionary algorithms are presented by elicitating a correspondence between their elements, which is summarized in Table 1.

In an AIN, the population of individuals is the set of B cells, and every B cell is usually represented as a real valued vector or a binary string. These individuals are stimulated by antigens and by other individuals, and the resulting stimulation level correspond to the subjective fitness of a coevolutionary algorithm.

Table 1. Parallel between coevolutionary algorithms and artificial immune networks. Notice that there is no equivalence in coevolutionary algorithms for the antigens and for the addition of new random cells to the population.

Coevolution	**Immune networks**
Individual	B cell
Population	Set of B cells (variable size)
-	Antigens
Subjective fitness	B cell stimulation level
Selection	Selection for cloning according to stimulation Natural death (metadynamics)
Reproduction	Cloning
Genetic operators	Somatic hypermutation
Competitive interaction	Suppression between B cells Competition for resources (in models with limited resources)
Cooperative interaction	The output of the algorithm is a set of detectors that cooperatively solve a problem Coestimulation between B cells
Spatiality	There is a notion of space (shape space), defined by the affinity measure. Detectors interact directly only with elements on their neighborhood in the shape space
-	B cell born (metadynamics)
Solution concept	Depends on the problem to be solved Not explicitly specified

An AIN's dynamics is similar to a steady state genetic algorithm, as a B cell can survive through several generations. A B cell may be selected for removal in metadynamics if it has a very low level of stimulation. Also, if it has a high level of stimulation, it undergoes cloning (asexual reproduction), and descendant cells suffer a mutation inversely related to the affinity with the current antigen (somatic hypermutation).

An AIN presents a mixed interaction between individuals; first, B cells interact in a cooperative way to solve a problem, that is, the result of the algorithm is the whole set of cells, rather than a single cell. Also, in models that consider limited resources, the B cells compete between them for such resources. Additionally, B cells interact between them to suppress or stimulate each other, which can be considered cooperative and competitive behavior respectively. Artificial models usually define a stimulation function and a suppression function, both of which depend on the affinity of a detector with its neighbors.

Individuals interact with each other and with antigens according to their affinity, which correspond to their similarity (in most cases) or complementarity. In other words, they interact with each other if they are neighbors in the shape space[2]. For real valued vectors the affinity metric is typically the Euclidean distance or a function of it . In the case of strings the Hamming distance, the edit distance, or a function of them could be used.

3.3 Differences Between AINs and Coevolution

Although AINs present several similarities with coevolutionary algorithms, they also present some particularities that make them different from coevolutionary algorithms.

In AINs, interaction is defined spatially, not as a way to improve an algorithm performance or to promote diversity, but because the model is inherently spatial. Also, a memory mechanism is inherent to immune networks as a result of its dynamics, while several coevolutionary algorithms implement some kind of memory in order to avoid cyclic dynamics, but as a external mechanism.

Additionally, immune network models are self regulatory, which means that they are free to vary population size, but this has to be done in a reasonable way, since there has to be an equilibrium state instead of an explosion in the number of cells.

Regarding solution concepts, there is no clear solution concept related to immune networks. Though the purpose of the algorithm is usually described, the properties a solution should have are rarely specified. Given the output of an AIN algorithm, clustering of the data is usually accomplished by connectivity of the detectors [7](each connected component of the network is a cluster), or by applying MST algorithm to the detector set[2].

In the next section, a solution concept for artificial immune network algorithms is proposed.

[2] A physical space in which B cells move (such as the blood torrent) is never considered.

4 A Solution Concept for Artificial Immune Networks

As discussed above, the solution concept formalism may be useful for characterizing AINs. Specifically, it could help to define performance measures, devise stopping criteria, and to refine algorithm dynamics in order to reach a desired solution. Therefore, here the purpose of the immune network model, its input and its output are characterized.

4.1 Characterization of the Solution Concept

Although AIN models have been adapted to perform a variety of tasks, here the purpose of AINs will be restricted and stated in machine learning terminology, in order to provide a precise definition of the algorithm and a unified framework to compare different algorithms. This solution concept is based on some concepts, such as probability density estimation, which will be specified later. An AIN algorithm is restricted to fit these criteria:

- An AIN algorithm performs unsupervised learning: The algorithm does not receive information about the labels of incoming data.
- An AIN algorithm performs semi-parametric probability density estimation: The output of the algorithm provides a model of the data, by representing the probability distribution of the training data.

Therefore, for a set of antibodies produced by an algorithm to be considered a solution, it must satisfy the following criteria:

- Equilibrium: An AIN is in equilibrium if it keeps its structure without change (or with a small change) indefinitely (or for a given period of time), given that there is not external stimuli. This criterion is related to the AIN memory mechanism.
- High antigen set modeling capability: AIN's cells should represent the distribution of antigens in the shape space. In other words, the probability distribution that it estimates should be as close as possible to the one that generated the antigens.

Therefore, the solution concept is defined as a subset of equilibrium states which satisfy these criteria.

Next, a characterization of AINs as probability density estimators is presented, as well as the construction of such probability density function, and measures associated to the proposed solution concept.

4.2 AINs as Semi-parametric Probability Density Estimators

Probability density estimation methods try to estimate a probability distribution P based on a set of samples of it. These methods can be classified into three categories: parametric, non-parametric and semi-parametric methods [1].

In parametric learning the data is supposed to be generated from some probability distribution, and the learning algorithm tries to find suitable values for the parameters of such distribution in order to fit the data.

In contrast, in non-parametric models an a priori distribution is not assumed. The model is built by associating a kernel function (usually Gaussian) to each point of the training data. Then, the model is constructed as a sum of such kernels. These methods rely on the fact that any probability distribution can be approximated as a mixture of Gaussians (see Fig.1). However, they require to store all the training data, and their use is computationally demanding.

In semi-parametric methods a set of kernels is obtained, but they do not correspond to the whole set of the training data. Only a few kernels are generated, which represent a model of the data. In this case as in the parametric case, parameters of the kernels are selected in order to fit the data in a better way.

AIN algorithms aim to obtain a set of B cells located in regions with high concentration of antigens. Then, if each B cell is assigned a Gaussian kernel, the generated probability distribution mimics the probability density of appearance of new antigens in a given region. Therefore, AINs algorithms can be considered semi-parametric density estimation methods, which try to model the probability density distribution of the antigens in the shape space, using a few number of kernels, each one associated to a B cell.

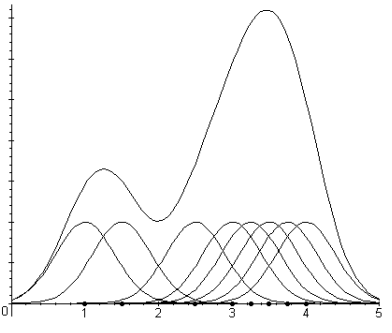

Fig. 1. Mixture of Gaussian kernels: The set of kernels generate the probability function shown. Notice that all Gaussian kernels have the same standard deviation and the same weight. However, in general, they can have different values for the standard deviation, and each kernel can have a different weight.

4.3 Antibody Set as a Model of Input Data

Input data will be assumed to be a set (training set TS) of n-dimensional real valued vectors (ag_i, $i \in \{1...N\}$), which are assumed to come from an unknown probability distribution P. The data is unlabeled, and may include noise.

After antigen exposure, the population of antibodies reaches a configuration (not necessarily static), which is the output of the algorithm. Such output is a set of detectors (antibodies ab_i, $i \in \{1...M\}$), along with the number of cells

or concentration of each clone (c_i, $i \in \{1...M\}$), for the algorithms that use the ARB representations), or the stimulation, for algorithms that consider individual antibodies. The detectors are also n-dimensional real valued vectors.

An antibody set is interpreted as a probability distribution in this way:

- Each detector ab_i is the center of a Gaussian kernel $k_i(x) \sim N(\mu_i, \sigma^2)$, where $\mu_i = ab_i$
- The standard deviation of the Gaussian kernel is the radius associated with the detector.
- The weight of each kernel is either the concentration (number of resources, for ARBs) or the stimulation level (for models representing individual B cells)

Accordingly, an antibody set is then interpreted as a probability distribution as follows:

$$\hat{P}(x) = (\sum_{i=1}^{M} k_i(x)c_i)/C, \text{ where } C = (\sum_{i=1}^{M} c_i)$$

4.4 AIN Modeling Capability

A measure used to calculate how well a probability distribution fits the data is the Likelihood. The likelihood expresses the probability that a dataset had been generated by a given probability distribution.

Therefore, the likelihood of $\hat{P}$ over the antigens will be used as a measure of how well a set of antibodies models the antigen set. For numerical reasons, it is better to consider the log-likelihood, which is defined as

$$L = \sum_{j=1}^{N} ln(\hat{P}(ag_j)) = \sum_{j=1}^{N} ln(\sum_{i=1}^{M} k_i(ag_j)c_i)$$

A high likelihood implies that the difference between the distributions P (the probability distribution from which antigens are sampled) and $\hat{P}$ (the estimated probability distribution) is expected to be minimum. Therefore, an AIN will be required to present a high likelihood over the antigens in order to be considered a solution.

4.5 Equilibrium

Memory requires that once a pattern is found in the data, it will not vanish later. Therefore, equilibrium is a requirement for memory.

Equilibrium implies that the likelihood of the network over the training data and the number of detectors do not vary significantly through time.

Thus, equilibrium can be tested by letting the AIN iterate without antigen presentation, and monitoring the number of network's cells and its likelihood over the data.

5 Exploratory Experimentation

Some preliminary experimentation was carried out in order to evaluate the antigen set modeling capability requirement of the solution concept.

The aiNet model, proposed by de Castro [2], was used to measure likelihood of the network using two synthetic datasets (the two spiral dataset and a two cluster dataset), and the IRIS dataset. Results are shown in figures 2, 3 and 4, respectively. In all cases, the parameters of the algorithm were set as follows: suppression threshold was set to 0.7, n=4, N=10, and qi=0.1. For the probability estimation, the standard deviation was taken as the suppression threshold, and all the kernels were assigned the same weight.

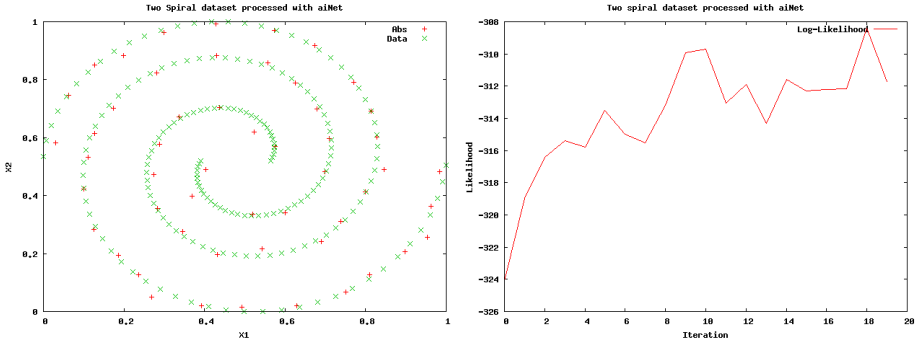

Fig. 2. Output and Likelihood evolution of the aiNet algorithm over the two spirals dataset (190 items)

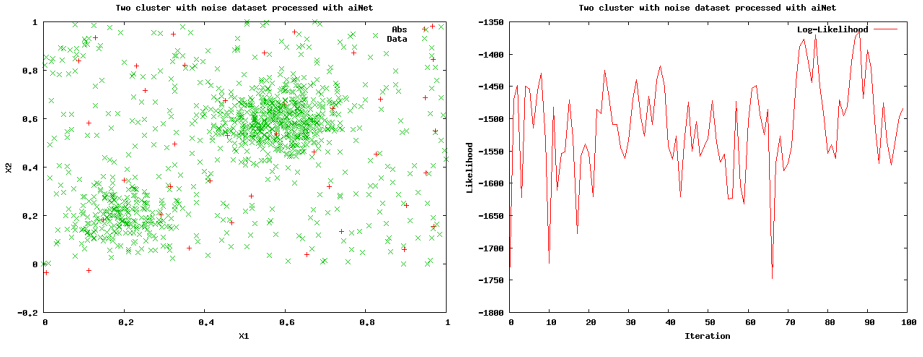

Fig. 3. Output and Likelihood evolution of the aiNet algorithm over the two clusters dataset (1018 items)

Results for the two spiral and the IRIS dataset show how the likelihood of the network increases along iterations, which indicates that it is a good measure of the algorithm progress. Also, results from the IRIS dataset show that the likelihood reaches a stable value about iteration 20, and later there is no further improvement in the likelihood. This suggest the use of the likelihood as a measure of convergence, and, therefore, as a stopping criterion: a stable value after several iterations suggest that the algorithm has converged.

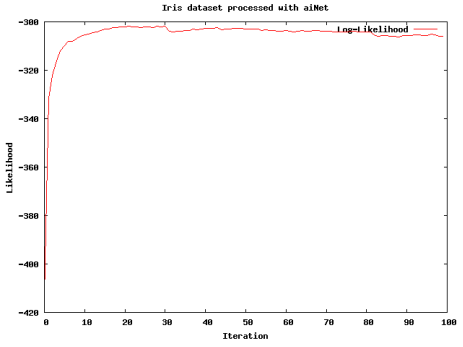

Fig. 4. Likelihood evolution of the aiNet algorithm over the IRIS dataset (150 items)

In the case of the two cluster dataset, the resulting configuration of the network does not resemble the distribution of input data. Therefore, the algorithm has failed to find an appropriate model of the data. It should be mentioned that the parameters of the AIN were set to provide such output. It is important to notice that the likelihood of the network does not increase throughout time, instead it shows an erratic behavior. Such behavior is an indicator of problems in the learning process, and suggest that the likelihood can be used to monitor the algorithm's performance.

6 Conclusions

In this paper, a formal relation between coevolutionary algorithms and artificial immune networks was established. It was shown that coevolutionary algorithms and AINs have many elements in common, but also that AINs present some characteristics that are not present in coevolutionary algorithms, such as variable size population, self regulatory behavior, inherent memory and spatiality.

Following Ficici's observations about coevolutionary dynamics, a solution concept for AINs was defined. Particularly, AINs are considered as a semi-parametric technique for probability density estimation: they estimate the probability function from which antigens are assumed to be sampled.

The proposed solution concept required the probability function estimated by the algorithm to have optimal likelihood over the training data; it also expects that the structure of the network remain stable in absence of antigens.

Network's likelihood was tested in the aiNet algorithm, and preliminary experimental results showed that it increased as the learning process took place, suggesting its use to monitor the algorithm's progress. Additionally, highly oscillatory likelihood over time was shown to indicate problems in the learning process.

A solution concept is useful to analyze the behavior of an AIN algorithm: results showed that it can be used to monitor algorithm's performance, and accordingly, to formulate stopping criteria. In addition, a solution concept may help to develop new algorithms or to improve existing ones.

As semi-parametric probability estimators, AINs have the advantage that they implicitly find the suitable number of required kernels.

Future work will focus on analyzing convergence properties of current AIN algorithms based on the proposed solution concept. Also, analytic work will be undertaken in order to relate immune network dynamics to the regularization theory, which attempts to limit the complexity of machine learning algorithms in order to achieve better generalization.

References

1. Bishop, C.M.: Neural networks for pattern recognition. Oxford University Press, Oxford, UK (1996)
2. de Castro, L.N., Von Zuben, F.J.: An evolutionary immune network for data clustering. In: França, F.M.G., Ribeiro, C.H.C. (eds.) SBRN, pp. 84–89. IEEE Computer Society, Los Alamitos (2000)
3. de Castro, L.N., Timmis, J.: Artificial Immune Systems: A New Computational Approach. Springer, London. UK (2002)
4. Ficici, S.G.: Solution Concepts in Coevolutionary Algorithms. PhD thesis, Computer Science Department. Brandeis University, USA (May 2004)
5. Galeano, J.C., Veloza-Suan, A., González, F.A.: A comparative analysis of artificial immune network models. In: GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation, pp. 361–368. ACM Press, New York, NY, USA (2005)
6. Jerne, N.K.: Towards a network theory of the immune system. Annals of Immunology 125C, 373–389 (1974)
7. Knight, T., Timmis, J.: Aine: An immunological approach to data mining. In: Cercone, N., Lin, T., Wu, X. (eds.) IEEE International Conference on Data Mining, pp. 297–304. IEEE, San Jose, CA. USA (2001)
8. Mitchell, M., Thomure, M.D., Williams, N.L.: The role of space in the success of coevolutionary learning. In: Artificial Life X: Proceedings of the Tenth International Conference on the Simulation and Synthesis of Living Systems. International Society for Artificial Life, pp. 118–124. The MIT Press(Bradford Books), Cambridge (2006)
9. Perelson, A.S., Weisbuch, G.: Immunology for physicists. Reviews of Modern Physics 69, 1219 (1997)
10. Potter, M.A., De Jong, K.A.: The coevolution of antibodies for concept learning. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) Parallel Problem Solving from Nature - PPSN V. LNCS, vol. 1498, pp. 530–539. Springer, Heidelberg (1998)
11. Wiegand, R.P., Sarma, J.: Spatial embedding and loss of gradient in cooperative coevolutionary algorithms. In: Yao, X., Burke, E.K., Lozano, J.A., Smith, J., Merelo-Guervós, J.J., Bullinaria, J.A., Rowe, J.E., Tiňo, P., Kabán, A., Schwefel, H.-P. (eds.) Parallel Problem Solving from Nature - PPSN VIII. LNCS, vol. 3242, pp. 912–921. Springer, Heidelberg (2004)