The Abstract Immune System Algorithm

José Pacheco[1⋆] and José Félix Costa[2,3]

[1] Departamento de Informática, Faculdade de Ciências e Tecnologia
Universidade Nova de Lisboa,
Quinta da Torre, 2829-516 Caparica, Portugal
jddp@di.fct.unl.pt
[2] Department of Mathematics, Instituto Superior Técnico
Universidade Técnica de Lisboa
Lisboa, Portugal
fgc@math.ist.utl.pt
[3] Centro de Matemática e Aplicações Fundamentais do Complexo Interdisciplinar
Universidade de Lisboa
Lisbon, Portugal

Abstract. In this paper we present an *Abstract Immune System Algorithm*, based on the model introduced by Farmer *et al*, inspired on the theory of Clonal Selection and Idiotypic Network due to Niels Jerne. The proposed algorithm can be used in order to solve problems much in the way that Evolutionary Algorithms or Artificial Neural Networks do. Besides presenting the Algorithm itself, we briefly discuss its various parameters, how to encode input data and how to extract the output data from its outcome. The reader can do their own experiments using the workbench found in the address `http://ctp.di.fct.unl.pt/~jddp/immune/`.

1 Introduction

Biological studies have always constituted a large pool of inspiration for the design of systems. In the last decades, two biological systems have provided a remarkable source of inspiration for the development of new types of algorithms: neural networks and evolutionary algorithms.

In recent years, another biological inspired system has attracted the attention of researchers, the immune system and its powerful information processing capabilities (e.g., [20, 10]). In particular, it performs many complex computations in a highly parallel and distributed fashion. The key features of the immune system are: pattern recognition, feature extraction, diversity, learning, memory, self-regulation, distributed detection, probabilistic detection, adaptability, specificity, etc. The mechanisms of the immune system are remarkably complex and poorly understood, even by immunologists. Several theories and mathematical models have been proposed to explain the immunological phenomena [18, 16]. There is also a growing number of computer models to simulate various components of the immune system and the overall behaviour from the biological point

⋆ Corresponding author

of view [7, 9]. Those approaches include differential equation models, stochastic differential equation models, cellular-automata models, shape-space models, etc.

The models based on the immune system principles, such as the Clonal Selection Theory [6, 12], the immune network model [15, 8], or the negative selection algorithm [14], have been finding increasing applications in science and engineering [9]: computer security, virus detection, process monitoring, fault diagnosis, pattern recognition, etc.

Although the number of specific applications confirms the interest and the capabilities of this principles, the lack of a general purpose algorithm for solving problems based on them contrasts with the major achievements with other Biologically inspired models. In this paper we present a modest *Abstract Immune System Algorithm*. Our paper is focused on describing the algorithm itself and how to use it.

2 The Biological Model

The immune system is a very complex system with several functional components [17, 13]. It is constituted by a network of interacting cells and molecules which recognize foreign substances. These foreign substances are called *antigens*. The molecules of the immune system that recognize antigens are called *antibodies*. An antibody does not recognize an antigen as a whole object. Instead, it recognizes small regions called *epitopes*. An antibody recognizes an antigen if it binds to one of its epitopes. The binding region of an antibody is called the *paratope*. The strength and the specificity of the interaction between antibody and antigen is measured by the affinity of the interaction. The affinity depends on the degree of complementarity in shape between the interacting regions of the antibody and the antigen. A given antibody can typically recognize a range of different epitopes, and a given epitope can be recognized by different antibody types. Not only do antibodies recognize antigens but they also recognize other antibodies if they have the right epitope. An epitope characteristic for a given antibody type is called an *idiotope*. Antibodies are produced by cells called *B-lymphocytes*. B-lymphocytes differ in the antibodies that they produce. Each type of antibody is produced by a corresponding lymphocyte which produces only this type of antibody. When an antibody on the surface of a lymphocyte binds another molecule (antigen or other antibody), the lymphocyte is stimulated to clone and to secrete free antibodies. In contrast, lymphocytes that are not stimulated die after days. Thus, a selection process is at work here where those antibodies that are stimulated by antigens or antibodies are amplified, while the other antibodies die out.

There exist several theories to explain the dynamics of the immune system. One of the most popular theories is the *immune network model* proposed by Niels Jerne [15]. Jerne hypothesized that the immune system is a regulated network of molecules and cells that recognize one another even in the absence of antigen. Such networks are often called *idiotypic networks* which present a mathematical framework to illustrate the behaviour of the immune system. His theory is

modeled with a system of differential equations which simulates the dynamics of lymphocytes, the increase or decrease of the concentration of lymphocyte clones and the corresponding immunoglobins. The idiotypic network hypothesis is based on the concept that lymphocytes are not isolated, but communicate with each other through interaction among antibodies. Jerne suggested that during an immune response antigens will induce the creation of a first set of antibodies. These antibodies would then act as antigens and induce a second set of *anti-idiotypic* antibodies. The repetition of this process produces a network of lymphocytes that recognize one another. With this hypothesis Jerne explains the display of memory by the immune system.

Other important issues are the incorporation of new types in the immune network and the removal of old types. The autoregulation of the immune system keeps the total number of different types roughly constant. Although new types are created continously, the probability that a newly created type is incorporated in the immune system is different for different types. In natural evolution, the creation of new individuals result from the mutation and crossover of the individuals in the population. In the immune system, new antibody types are created preferably in the neighbourhood of the existing high-fit antibody types. The biased incorporation into the immune network of randomly created species is called the *recruitment strategy* [5, 3, 4].

Also fundamental for the present formulation is clonal selection theory [6, 12]. This theory states that there is a bisimilar relation between lymphocytes and receptors, hence each kind of lymphocyte has only one kind of receptor. To explain the predominance or the disappearing of certain kinds of lymphocytes, the theory assumes a highly diverse initial population, composed by randomly distributed lymphocytes. The recognition of antigens produces a reaction of lymphocytes are stimulating their reproduction, through the production of clones. The measure of the adaptability of lymphocytes, is proportional to the intensity of the reaction, giving more chances of survival to those kinds of lymphocytes more stimulated on a given environment. This natural selection process together with the high rate of mutation found on the reproduction stage, induces an increase of the concentration of the lymphocytes with the highest molecular affinity with the form they try to eliminate.

3 The Algorithm

The algorithm presented here follows closely the model proposed by Farmer *et al* [11] for the dynamics of a system based on the idiotypic network model. The dynamics of the model is described using a set of differential equations of the form:

$$\dot{x}_i = c \left[\sum_{j=1}^{N} m_{ji} x_j x_i - k_1 \sum_{j=1}^{N} m_{ij} x_i x_j + \sum_{j=1}^{n} m^*_{ji} y_j x_i \right] - k_2 x_i$$

where N is the number of antibody types, with concentrations, respectively, x_1, x_2, ..., x_N, and n is the number of antigen types with concentrations y_1, y_2, ...,

y_n; m^*_{ji} correspond to the affinity between the paratope of the antibody i and the epitope of the antigen j; m_{ji} correspond to the affinity between the paratope of the antibody i and the epitope of the antibody j.

The first term represents the stimulation of the paratope of an antibody of type i by the epitope of an antibody of type j. The second term represents the suppression of the antibody of type i when its epitope is recognized by the paratope of type j. The constant k_1 represents the possible inequality between stimulation and suppression. The third term models the stimulation provided by the recognition of the antigen j (with concentration y_j) by the antibody of type i (with concentration x_i). The final term models the tendency of cells to die in the absence of any interaction, at a rate determined by k_2. The parameter c is a rate constant that depends on the number of collisions per unit of time and the rate of antibody production stimulated by a collision.

Since we want to model the evolution of a population of potential solutions of a problem, we consider the following set of equations:

$$\dot{x}_i = c\ [\ \sum_{j=1}^{N}\ m_{ji}x_jx_i - k_1\ \sum_{j=1}^{N}\ m_{ij}x_ix_j\ ] - k_2x_i,$$

ignoring the antigens.

With some further simplification we get the equation:

$$\dot{x}_i = [\ c\sum_{j=1}^{N}\ (m_{ji} - k_1m_{ij})x_j - k_2\ ]\ x_i$$

and finally:

$$\dot{x}_i = (\sum_{j=0}^{N}\ n_{ij}x_j)x_i$$

where $n_{ij} = c\ (m_{ji} - k_1m_{ij})$, for $j \neq 0$, $n_{0j} = 0$, for all j, $n_{i0} = -k_2$, for $i \neq 0$, and $x_0 = 1$ by convention.

Since this system of differential equations is not integrable using analytic methods, the algorithm uses the finite difference method to update the concentrations of the various antibody types. Therefore, the concentration of each antibody is approximated, on each step of the simulation, using the update rule:

$$x_i(t + h) = x_i(t) + h\dot{x_i}(t)$$

E.g.,

$$x_i(t + h) = x_i(t) + h\ x_i(t)\ (\ \sum_{j=0}^{N}\ n_{ij}x_j(t)\ )$$

If $k_1 = 1$ (equal stimulation and suppression) and $k_2 > 0$, then every antibody type eventually dies due to the damping term. Taking $k_1 < 1$ favors the formation of reaction loops, since all the numbers of a loop can gain concentration and

thereby fight the damping term. As N increases, so does the length and number of loops. The robust properties of the loops allow the system to remember certain states even when the system is disturbed by the introduction of new types.

Along with the three parameters (c, k_1 and k_2), which allow the tuning of the algorithm, two others are necessary for the specification of the algorithm. We need to set a *death threshold* and a *recruitment threshold*. The first establish the minimum concentration under what a given antibody type is eliminated from the population. The second indicates the minimum concentration above what a given antibody is sufficiently stimulated to initiate the recruitment process. Other parameters might be necessary depending on the recruitment process used on a given implementation.

We can now formulate the *Abstract Immune System Algorithm* as the following steps:

```
Randomly initialize initial population
Until termination condition is met do
   Update concentrations using equation formula
   Eliminate antibodies under death threshold
   Recruitment of new antibodies
```

As usual with this kind of algorithms, several types of termination conditions can be considered (time, number of generations, stability, maximum concentration, ...). As for the recruitment process, there are also many approaches that can be followed. The simplest one would be just clone and mutate the antibodies with a probability proportional to their concentrations, but we can use a scheme where crossover and other evolutionary processes are involved.

4 Problems and Results

In order to use the Algorithm to solve a given problem we need to know how to encode it and how to extract the result once the algorithm halts.

Each antibody represents a possible solution to the problem. Although the usual binary representation is always possible, sometimes it is preferable to use a different representation in order to avoid invalid solutions.

Given an encoding, the second and probably the most important thing to do is the function that computes the affinity between one paratope and one epitope. For this function to be effective it must take into account not only the affinity between the two antibodies but also enhance this value according to the relative quality of the idiotop as a possible solution to the problem. This function corresponds to the m matrix.

A third thing that need to be established is the method of recruitment used by the algorithm. The method can be as simple as clone and mutate antibodies with probability proportional to their concentrations, but it can also include other methods such as different types of crossover or any other evolutionary

method. If the newly generated antibody doesn't exist in the population, then it is introduced with a fixed concentration. This is the way to express diversity.

As we might expect several possible encodings exist for a given problem and the choice will greatly influence the results obtained. Besides the different parameters, other factors that can influence the result are the initial population (although the algorithm states that it is randomly generated we can consider the situation where the initial population is carefully chosen), its dimension (N), etc.

The outcome of the algorithm will be the individual with the highest concentration of the population. Since the concentration is closely related with the probability of using a given antibody, it seems reasonable to take as answer to the problem the most probable potential answer found. Several other schemes could be considered depending on the problem itself.

5 The Probabilistic Computational Model

We consider probabilistic algorithms as in [2], chapter 6. Intuitively it is easy to recognize our Algorithm as probabilistic. The factors of uncertainty are the randomly generated initial population and the selection of antibodies for recruitment. Each of these factors can be described as probabilistic guesses. In this section we will address the problem of describing the Algorithm in a probabilistic format. First we will present its formal description and then we will analyze the probability related to each step of the process.

Consider a population of N antibody types. Each antibody is represented as a sequence of size n. In generation k, the bit j of the antibody i, is a random bit variable y_{ij}^k. The concentration of antibody i in generation k is the variable x_i^k, with $0 \leq x_i^k \leq 1$. We assume normalization: $\sum_{i=1}^{N} x_i^k = 1$, for all k.

In a rough analysis of our Algorithm, we introduce a few simplifying assumptions.

Assumption 1 *We assume a unique optimal solution so that each bit of the antibody type – solution has an unique correct value.*

Assumption 2 *We assume a surrogate affinity measure that is proportional to the antibody quality. The quality is measured by the number of correct bits in the antibody sequence.*

Although these assumptions are not always true, since many problems have multiple optima and the affinity measure can take any form, they are reasonable in many cases and hopefully will lead to new insights. By Assumption 1, there is a single correct value for each bit in the antibody type – solution. Let b_{ij}^k be the bit that tells us if y_{ij}^k is the correct value. Let $S_i^k = \sum_{j=1}^{n} b_{ij}^k$ be the number of correct values in antibody i in the generation k. By Assumption 2, the affinity function is proportional to S_i^k. The best element of the population is A^k that corresponds to $\max_i S_i^k$. Given a generation, with affinities $\{S_i^{k-1}\}_{1 \leq i \leq N}$, with $k > 0$, we can construct the next generation by successively updating and normalizing antibody

concentrations, removing the antibody types with concentrations under the *death threshold* and incorporating newly generated antibodies. The *Abstract Immune System Algorithm* can be formally stated as follows:

(Initialize) Generate uniformly y_{ij}^0, for every $1 \leq i \leq N$, $1 \leq j \leq n$. Assign x_i^0 to its initial value. Calculate all S_i^0 and A^0. $k \leftarrow 1$.

(Iterate) While $A^{k-1} < n$ do

(Update) Use the rule

$$x_i^k = x_i^{k-1} + c \int_{k-1}^{k} x_i(t) \sum_{j=1}^{N} n_{ij} x_j(t) dt$$

to update the concentrations of the antibody types. Normalize the concentrations.

(Eliminate) Remove all antibody types with concentrations x_i^k less than the *death threshold*. Update the value of N.

(Recruitment) Randomly select for recruitment, with probabilities x_i^k, antibody types with concentrations greater than the *recruitment threshold*. Mutate the selected antibody types (modifying one bit of each) introducing the newly created antibodies into the population. Update the value of N.

(Evaluate) Evaluate each S_i^k. Determine A^k.

(Increment) $k \leftarrow k + 1$.

(end iteration)

Some steps of the Algorithm display a completely deterministic behaviour. *Update* will only affect the distribution of concentrations within the antibody types of the population. This will only influence the choice of an antibody for deletion and for recruitment. *Eliminate* will affect the population, but since an antibody eligible for recruitment could never be eligible for deletion it will not affect the *Recruitment* step. We will then concentrate in the two steps that display probabilistic behaviour.

For the initial population, y_{ij}^0 is generated uniformly. The probability of choosing a correct bit is always $1/2$ for each bit. Also b_{ij}^0 is like tossing n coins. By construction, y_{ij}^0 and b_{ij}^0 are independent variables. Hence S_i^0 has binomial distribution $B(n, 1/2)$. Further, all S_i^0 are independent from each other, though clearly not independent of the corresponding y_{ij}^0 and b_{ij}^0. The average affinity of the initial population is proportional to $E[S_i^0] = n/2$. With N independent members of the population, we have $P(A^0 \leq J) = P(\max_i S_i^0 \leq J) = P(S_i^0 \leq J, \forall i) = (F(J))^N$ where $F(J)$ is the cumulative distribution function of $B(m, 1/2)$, from

0 to J. The expectation value of A^0 is given by the expression

$$E[A^0] = \sum_{J=0}^{n-1} [1 - (F(J))^N].$$

In generation 0 the members of the population are independent. Subsequent generations depend on the previous ones. However, their correlation is greatly reduced by virtue of the distribution of affinities and for sufficiently large N.

The first part of the recruitment process is the selection of antibody types determined by the concentrations. Only antibody types with concentrations above the *recruitment threshold* are eligible for reproduction. The process of *Recruitment* will increase the size of the population. Given an antibody type p selected at random with affinity S_p^{k-1}, we derive the probability distribution of the affinity S_i^k of the offsprings. In this rough analysis we make one more assumption.

Assumption 3 *Given that the parent have J correct bits, the location of the correct bits is distributed uniformly and the correct bits are independent of each other.*

Assumption 3 will be true in early generations, but becomes less accurate as the population converges. The impact of this assumption will be judged empirically. Given the selected antibody type, the mutated bit will either change from an incorrect value to a correct one or change from a correct to an incorrect value. Then for the offspring we have

$$S_i^k = \sum_{j \neq l}^{n} b_{ij}^k + w_l$$

where w_l is a Bernoulli variable corresponding to probability $1/2$. The maximum value of J can take is given when all the correct bits remain unchanged and the bit selected mutates to a correct value. In this situation we have $S_i^k = S_i^{k-1} + 1$. Under the Assumption 3 we have

$$P(S_i^k = J | S_p^{k-1} = J') = \binom{n}{J} (\frac{J'}{n})^J (1 - \frac{J'}{n})^{(n-J)}$$

For a sufficiently large N, we may approximate $P(S_i^k \leq J_i, 1 \leq i \leq N) \approx \prod_{i=1}^{N} P(S_i^k \leq J_i)$. According to our assumptions, we have

$$P(S_i^k \leq J) \approx \sum_{I=0}^{n} P(S_i^k \leq J | S_p^{k-1} = I) P(S_p^{k-1} = I)$$

The first factor of the sum can be computed from the equation above for $P(S_i^k = J | S_p^{k-1} = J')$. The second factor is given as the inductive step. Then

$$E[S_i^k] = \sum_{J=0}^{n-1} [1 - P(S_i^k \leq J)^N].$$

Since $P(A^k \leq J) \approx [P(S_p^k \leq J)]^N$, we get $E[A^k] \approx \sum_{J=0}^{n-1}[1 - P(A^k \leq J)]$. To compute $E[A^k]$, we compute the anchor distribution of S_i^0 as given before, and then, inductively, we compute the distributions of S_i^k.

6 Examples

In this section we present two of the examples used to test the Algorithm. A workbench of classes/interfaces in Java was programmed. The source code, its documentation and the examples (to be tested by the reader) can be found at the address `http://ctp.di.fct.unl.pt/~jddp/immune/`.

The first example selected is a classical maximization problem. The aim of this example is to show how to use the Algorithm in its simplest form to solve a problem that is very well studied for other Biologically inspired algorithms. We considered the power function $f(x) = (x/c)^n$, where c is a normalizing constant (with $n = 10$).

We considered a binary representation with 30 bits per antibody corresponding to a large search space. Then we had to define an affinity function. As suggested in Section 4, we considered two factors to compute this function. The first factor represents the quality of the antibody, given by the function f we aim to optimize. The second factor represents the correlation between an antibody and any other antibody of the population, for this we used the Hamming distance between the two antibody strings u and v of size m

$$\mathcal{H}(u, v) = \sum_{i=1}^{m} u_i \ (1 - v_i) + \sum_{i=1}^{m} v_i \ (1 - u_i).$$

The final affinity function is

$$\mathit{Affinity}(u, v) = f(u) \times \mathcal{H}(u, v)$$

The third aspect is the recruitment method. We add to the population mutated clones of the antibodies with concentration above the *recruitment threshold*.

The Algorithm was used with different initial population sizes and various parameters and, most of the times, it was capable of finding a near optimal solution in only a few generations.

As second example we considered the *Iterated Prisoner's Dilemma*. The aim of this example is to show the use of the Algorithm with a less trivial problem, and the use of more sophisticated antibody representation and affinity function.

The prisoner's dilemma is a classic problem of conflict and cooperation [19, 1]. In its simplest form each of two players has a choice of cooperating with the other or defecting. Depending on the two players' decisions, each receives payoff according to a payoff matrix. When both players cooperate they are both rewarded at an equal, intermediate level (reward, R). When only one player defects, he receives the highest level of payoff (temptation, T), while the other player gets the sucker's just deserts (sucker, S). When both players defect they each receive an intermediate penalty (P). The prisoner's dilemma has often been

cited as a simple yet realistic model of the inherent difficulty of achieving cooperative behaviour when rewards are available for the successful miscreant. The problem is called the prisoner's dilemma because it is an abstraction of the situation felt by a prisoner who can either cut a deal with the prosecutor and thereby rat on his partner in crime (defect) or keep silent and therefore tell nothing of the misdeed (cooperate).

		Player 2	
		Cooperate	Defect
Player 1	Cooperate	R=6, R=6	S=0, T=10
	Defect	T=10, S=0	P=2, P=2

Table 1: Prisoner's Dilemma Payoff Matrix.

The problem is made more interesting by playing it repeatedly with the same group of players, thereby permitting partial time histories of behaviour to guide future decisions. This so-called iterated prisoner's dilemma has drawn interest from game theorists for a while.

A strategy for the Iterated Prisoner's Dilemma is not obvious. The goal to the algorithm was to come up with the best possible strategy.

As ever, our first concern is to provide a representation for the possible solutions of the problem. To represent a strategy we will need to what to do next based on the history of past plays.

In our representation, to decide on a current move, a current player will look at the past three rounds before making a decision. If a single round can be represented by 2 bits: CC, CD, DC, DD, a three round memory requires 6 bits. By choosing to represent a cooperation (C) as a 0 and a defect (D) as a 1, a simple history pattern can be recorded. For instance, 10 00 01 will represent that Player 2 defected three rounds ago, they both cooperated two rounds ago, and Player 1 defected last round. In order to choose a current move, a Player must have a strategy for all possible previous three rounds. In other words, a Player must have 64 (2^6) different strategies on how to play. Also, a strategy consists of defecting or cooperating, which can be represented by a single bit (0 for cooperate and 1 for defect). These facts allow the creation of our bit string for the algorithm. Each bit represents a strategy for a given past three rounds. The index value for the bit string is directly computable from the 6-bit history, it is merely the conversion of the 6-bit history into the corresponding number. So, the representation of a strategy for the prisoner's dilemma is a 70-bit string. The first 64-bits are to be used to determine the current move based on past moves. The last 6 bits are to be used to determine the first move. Of course this representation can be generalized to consider the n previous rounds to make the decision, instead of just three. Note however that the size of the antibodies will increase exponentially.

Our second concern is to choose an affinity function. The obvious choice is to make different strategies play against each other and use the difference of the scores as the affinity. Obviously, the number of rounds we choose to play

will influence the accuracy of the affinity function. On the other hand, since the algorithm will test the affinity of each antibody against all other, choosing a large number of rounds will slow down the algorithm greatly. For this example we used and affinity function that will play 20 rounds.

Our last concern is to provide a recruitment method for the algorithm. As we did on the first example, as recruitment method we use cloning and mutation. A mutation function can easily be implemented by randomly change the value of a small number of bits.

Finally, as a way to improve the performance of the algorithm, we introduced in the initial population several antibodies representing well known proposed strategies, such as:

- **nice** – Player cooperates all the time;
- **mean** – Player defect all the time;
- **tit-for-tat** – Player cooperates for the first move, and the copies the other player's last move;
- **opp-tit-for-tat** – Player defects for the first move, and the copies the other player's last move;
- **cd** – Periodic behavior - plays $CDCDCD$
- **ccd** – Periodic behavior - plays $CCDCCD$
- **dcc** – Periodic behavior - plays $DDCDDC$
- **pavlov** – Player cooperates if both players played the same action in the last round.

Even with a random initial population the algorithm was able to discover strategies that beat the overall performance of the bf tit-for-tat strategy.

7 Conclusions and Future Work

We proposed a problem solving algorithm based on the model of the immune system of Farmer *et al*, referred as *Abstract Immune System Algorithm*. The Algorithm proved to be capable to solve small to moderate size problems in an efficient and reliable way.

Future work might concentrate on the study of variants of the Algorithm and the impact of its parameters. As with evolutionary algorithms, the *Abstract Immune System Algorithm* can be subject of variations, concerning either the order of operations or different recruitment strategies. Another area for further exploration is the experimentation of the algorithm in a larger set of problems and the study its adequacy. We might be concerned both with the quality of the results and the characterization of the class of problems best suited for the algorithm. The consideration of different representations, affinity functions and recruitment methods dealing with well known problems can prove of particular interest.

Acknowledgements. The authors are indebted to Jorge Orestes Cerdeira, Carlos Lourenço and Helder Coelho for detailed criticism about this work. José Félix

Costa also thanks to his previous students João Maças and Francisco Martins who started in 1995 this project with him.

References

1. Robert Axelrod. *The Complexity of Cooperation: Agent-Based Models of Competition and Collaboration*. Princeton Press, 1997.
2. J. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I*. Springer-Verlag, second edition, 1995.
3. Hugues Bersini. Reinforcement learning and recruitment mechanism for adaptive distributed control. Technical Report IR/IRIDIA/92-4, Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificielle, 1992.
4. Hugues Bersini and Gregory Seront. Optimizing with the immune recruitment mechanism. Technical Report TR/IRIDIA/93-6, Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificielle, 1993.
5. Hugues Bersini and Francisco J. Varela. The immune recruitment mechanism: A selective evolutionary strategy. Technical Report IR/IRIDIA/91-5, Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificielle, 1991.
6. Frank MacFarlane Burnet. *The Clonal Selection Theory of Acquired Immunity*. Cambridge University Press, 1959.
7. Franco Celada and Philip E. Seiden. A computer model of cellular interactions in the immune system. *Immunology Today*, 13(2):56–62, 1992.
8. Antonio Coutinho. The network theory: 21 years later. *Scandinavian Journal of Immunology*, 42:3–8, 1995.
9. Dipankar DasGupta, editor. *Artficial Immune Systems and Their Applications*. Springer-Verlag, 1998.
10. L.N de Castro and J. Timmis. *Artificial Immune Systems: A New Computational Approach*. Springer-Verlag, 2002.
11. J. Doyne Farmer, Norman H. Packard, and Alan S. Perelson. The immune system, adaptation, and machine learning. *Physica D*, 22:187–204, 1986.
12. Donald R. Forsdyke. The origins of the clonal selection theory of immunity as a case study for evaluation in science. *The FASEB Journal*, 9:164–166, Feb 1995.
13. W. John Herbert, Peter C. Wilkinson, and David I Stott. *The Dictionary of Immunology*. Academic Press, 4th edition, 1995.
14. Niels K. Jerne. The natural-selection theory of antibody formation. *Proceedings of The National Academy of Sciences USA*, 41:849–856, 1955.
15. Niels K. Jerne. Towards a network theory of the immune system. *Ann. Immunol. (Inst. Pasteur)*, 125(C):373–389, 1974.
16. Chris Lord. An emergent homeostatic model of immunological memory. Technical report, Carnegie Mellon Information Networking Institute, Feb 2002.
17. William E. Paul, editor. *Fundamental Immunology*. Raven Press, New York, 3rd edition, 1993.
18. Alan S. Perelson and Gérard Weisbuch. Immunology for physicists. *Reviews of Modern Physics*, 69(4):1219–1263, 1997.
19. William Poundstone. *Prisoner's Dilemma: John von Neumann, Game Theory and the Puzzle of the Bomb*. Doubleday Publishing, 1993.
20. A.O. Tarakanov, V.A. Skormin, and S.P. Sokolova. *Immunocomputing: Principles and Applications*. Springer-Verlag, 2003.