Learning and Optimization Using the Clonal Selection Principle

Leandro N. de Castro, *Member*, *IEEE*, and Fernando J. Von Zuben, *Member*, *IEEE*

Abstract — **The clonal selection principle is used to explain the basic features of an adaptive immune response to an antigenic stimulus. It establishes the idea that only those cells that recognize the antigens are selected to proliferate. The selected cells are subject to an affinity maturation process, which improves their affinity to the selective antigens. In this paper, we propose a computational implementation of the clonal selection principle that explicitly takes into account the affinity maturation of the immune response. The general algorithm, named CLONALG, is primarily derived to perform machine-learning and pattern recognition tasks, and then it is adapted to solve optimization problems, emphasizing multimodal and combinatorial optimization. Two versions of the algorithm are derived, their computational cost per iteration is presented and a sensitivity analysis with relation to the user-defined parameters is given. CLONALG is also contrasted with evolution strategies and genetic algorithms. Several benchmark problems are considered to evaluate the performance of CLONALG and it is also compared to a niching method for multimodal function optimization.**

Index Terms — **Artificial immune systems, clonal selection principle, evolutionary algorithms, optimization, pattern recognition.**

I. INTRODUCTION

Over the last few years, there has been an ever increasing interest in the area of *Artificial Immune Systems* (AIS) and their applications. Among the many works in this new field of research, we can point out [1]–[6]. AIS aim at using ideas gleaned from immunology in order to develop systems capable of performing a wide range of tasks in various areas of research.

In this work, we will review the clonal selection concept, together with the affinity maturation process, and demonstrate that these biological principles can lead to the development of powerful computational tools. The algorithm to be presented focuses on a systemic view of the immune system

The authors are with the School of Electrical and Computer Engineering (FEEC), State University of Campinas (UNICAMP), Campinas, SP, Brazil, (e-mail: {lnunes,vonzuben}@dca.fee.unicamp.br)

and does not take into account cell-cell interactions. It is not our concern to model exactly any phenomenon, but to show that some basic immune principles can help us not only to better understand the immune system itself, but also to solve complex engineering tasks.

The algorithm is initially proposed to carry out machine-learning and pattern recognition tasks and then adapted to solve optimization problems. No distinction is made between a B cell and its receptor, known as antibody, hence every element of our artificial immune system will be generically called antibody.

First, we are going to apply the clonal selection algorithm to a binary character recognition problem in order to verify its ability to perform tasks such as learning and memory acquisition. Then it will be shown that the same algorithm, with few modifications, is suitable for solving multimodal and combinatorial optimization. This work also contains a discussion relating the proposed clonal selection algorithm, named CLONALG, with well-known evolutionary algorithms.

The paper is organized as follows. In Section II we review the clonal selection principle and the affinity maturation process from an immunological standpoint. Section III characterizes clonal selection as an evolutionary process, and Section IV briefly discusses a formalism to model immune cells, molecules, and their interactions with antigens. Section V introduces the two versions of the proposed algorithm, while Section VI reviews the basics of evolutionary algorithms. In Section VII we present the performance evaluation of the proposed algorithm, including a comparison with a niching approach to perform multimodal optimization. Section VIII contains the sensitivity analysis of the algorithm with relation to the user-defined parameters, and Section IX discusses the main properties of the algorithm in theoretical and empirical terms. The work is concluded in Section X.

II. THE CLONAL SELECTION THEORY

When an animal is exposed to an antigen, some subpopulation of its bone marrow derived cells (B lymphocytes) respond by producing antibodies (Ab). Each cell secretes a single type of antibody, which is relatively specific for the antigen. By binding to these antibodies (cell receptors), and with a second signal from accessory cells, such as the T-helper cell, the antigen stimulates the B cell to proliferate (divide) and mature into terminal (non-dividing) antibody secreting cells, called plasma cells. The process of cell division (mitosis) generates a clone, i.e., a cell or set of cells that are the progenies of a single cell. While plasma cells are the most active antibody secretors, large B lymphocytes, which divide rapidly, also secrete antibodies, albeit at a lower rate. On the other hand, T

cells play a central role in the regulation of the B cell response and are preeminent in cell mediated immune responses, but will not be explicitly accounted for the development of our model.

Lymphocytes, in addition to proliferating and/or differentiating into plasma cells, can differentiate into long-lived B memory cells. Memory cells circulate through the blood, lymph and tissues, and when exposed to a second antigenic stimulus commence to differentiate into large lymphocytes capable of producing high affinity antibodies, pre-selected for the specific antigen that had stimulated the primary response. Fig. 1 depicts the clonal selection principle.

The main features of the clonal selection theory that will be explored in this paper are [7], [8]:

- Proliferation and differentiation on stimulation of cells with antigens;

- Generation of new random genetic changes, subsequently expressed as diverse antibody patterns, by a form of accelerated somatic mutation (a process called *affinity maturation*); and

- Elimination of newly differentiated lymphocytes carrying low affinity antigenic receptors.

A. Reinforcement Learning and Immune Memory

Learning in the immune system involves raising the relative population size and affinity of those lymphocytes that have proven themselves to be valuable by having recognized a given antigen. In our use of the clonal selection theory for the solution of practical problems, we do not intend to maintain a large clone for each candidate solution, but to keep a small set of best individuals. A clone will be temporarily created and those progeny with low affinity will be discarded. The purpose is to solve the problem using a minimal amount of resources. Hence, we seek high quality and parsimonious solutions.

In the normal course of the immune system evolution, an organism would be expected to encounter a given antigen repeatedly during its lifetime. The initial exposure to an antigen that stimulates an adaptive immune response is handled by a spectrum of small low affinity clones of B cells, each producing an antibody type of different affinity. The effectiveness of the immune response to secondary encounters is considerably enhanced by the presence of memory cells associated with the first infection, capable of producing high affinity antibodies just after subsequent encounters. Rather than 'starting from scratch' every time, such a strategy ensures that both the speed and accuracy of the immune response becomes successively greater after each infection. This is an intrinsic scheme of a *reinforcement learning strategy* [9], where the system is continuously improving the capability to perform its task.

To illustrate the adaptive immune learning mechanism, consider that an antigen Ag_1 is introduced at time zero and it finds a few specific antibodies within the animal (see Fig. 2). After a *lag* phase, the antibody against antigen Ag_1 appears and its concentration rises up to a certain level, and then starts to decline (*primary response*). When another antigen Ag_2 is introduced, no antibody is present, showing the specificity of the antibody response [10]. On the other hand, one important characteristic of the immune memory is that it is associative: B cells adapted to a certain type of antigen Ag_1 presents a faster and more efficient *secondary response* not only to Ag_1, but also to any structurally related antigen $Ag_1 + Ag_2$. This phenomenon is called *immunological cross-reaction*, or *cross-reactive response* [11]–[15]. This associative memory is contained in the process of vaccination and is called *generalization capability*, or simply *generalization*, in other artificial intelligence fields, like *neural networks* [16].

Some characteristics of the associative memories are particularly interesting in the context of artificial immune systems:

- The stored pattern is recovered through the presentation of an incomplete or corrupted version of the pattern; and

- They are usually robust, not only to noise in the data, but also to failure in the components of the memory.

By comparison with the primary response, the secondary response is characterized by a shorter lag phase, a higher rate, and longer persistence of antibody synthesis. Moreover, a dose of antigen substantially lower than that required to initiate a primary response can cause a secondary response.

Some authors [17], [18] suggested that long-lived B cells, which have responded to an antigenic previous exposure, remain in a small, resting state, and will play an important role in secondary antibody responses. These memory cells are disconnected, at least functionally, from the other cells, and memory is a clonal property, at least in the context of secondary responses, in which the clone size of an antigen specific cell has increased. Other authors [14] argued that whether memory cells are truly resting cells is a debatable issue and suggested that typical memory cells are semi-activated cells engaged in low-grade responses to persisting antigens.

It is important to remark that, under an engineering perspective, the cells with higher affinity must somehow be preserved as high quality candidate solutions, and shall only be replaced by improved candidates, based on statistical evidences. This is the reason why in the first version of our model we maintain a specific memory set as part of the whole repertoire.

As a summary, immune learning and memory are acquired through [19]:

- Repeated exposure to a pathogen;

- Affinity maturation of the receptor molecules (see next section);

- Low-grade chronic infection; and

- Cross-reactivity to endogenous and exogenous pathogens.

B. Somatic Hypermutation, Receptor Editing and Repertoire Diversity

In a T cell dependent immune response, the repertoire of antigen-activated B cells is diversified basically by two mechanisms: *hypermutation* and *receptor editing* [20]–[23].

Antibodies present in a memory response have, on average, a higher affinity than those of the early primary response. This phenomenon, which is restricted to T-cell dependent responses, is referred to as the *maturation of the immune response*. This maturation requires the antigen-binding sites of the antibody molecules in the matured response, to be structurally different from those present in the primary response.

Random changes are introduced into the genes responsible for the Ag-Ab interactions, and occasionally one such change will lead to an increase in the affinity of the antibody. It is these higher-affinity variants which are then selected to enter the pool of memory cells. Not only the repertoire is diversified through a hypermutation mechanism, but also mechanisms must exist such that rare B cells with high affinity mutant receptors can be selected to dominate the response. Those cells with low affinity or self-reactive receptors must be efficiently eliminated, become anergic (with no function) or be edited [20]–[22].

Recent results suggest that the immune system practices molecular selection of receptors in addition to clonal selection of lymphocytes. Instead of the expected clonal deletion of all self-reactive cells, occasionally B lymphocytes were found that had undergone receptor editing: these B cells had deleted their low affinity receptors and developed entirely new ones through V(D)J recombination [23].

Receptor editing offers the ability to escape from local optima on an affinity landscape. Fig. 3 illustrates this idea by considering all possible antigen-binding sites depicted in the *x*-axis, with the most similar ones adjacent to each other. The Ag-Ab affinity is shown on the *y*-axis. If it is taken a particular antibody (Ab_1) selected during a primary response, then point mutations allow the immune system to explore local areas around Ab_1 by making small steps towards an antibody with higher

affinity, leading to a local optima (Ab_1^*). Because mutations with lower affinity are lost, the antibodies can not go down the hill. Receptor editing allows an antibody to take large steps through the landscape, landing in a locale where the affinity might be lower (Ab_2). However, occasionally the leap will lead to an antibody on the side of a hill where the climbing region is more promising (Ab_3), reaching the global optimum. From this locale, point mutations can drive the antibody to the top of the hill (Ab_3^*). In conclusion, point mutations are good for exploring local regions, while editing may rescue immune responses stuck on unsatisfactory local optima.

In addition to somatic hypermutation and receptor editing, a fraction of newcomer cells from the bone marrow is added to the lymphocyte pool in order to maintain the diversity of the population. According to [24], from 5% to 8% of the least stimulated lymphocytes is replaced per cell generation, joining the pool of available antigen recognizing cells. This may yield the same result as the process of receptor editing, i.e., a broader search for the global optimum of the antigen-binding site.

1) The Regulation of the Hypermutation Mechanism

A rapid accumulation of mutations is necessary for a fast maturation of the immune response, but the majority of the changes will lead to poorer or non-functional antibodies. If a cell that has just picked up a useful mutation continues to be mutated at the same rate during the next immune responses, then the accumulation of deleterious changes may cause the loss of the advantageous mutation. The selection mechanism may provide a means by which the regulation of the hypermutation process is made dependent on receptor affinity. Cells with low affinity receptors may be further mutated and, as a rule, die if they do not improve their clone size or antigenic affinity. In cells with high-affinity antibody receptors, however, hypermutation may become inactive [17], [20], generally in a gradual manner.

III. AN EVOLUTIONARY SYSTEM

The clonal selection functioning of the immune system can be interpreted as a remarkable microcosm of Charles Darwin's law of evolution, with the three major principles of repertoire diversity, genetic variation and natural selection [25]. Repertoire diversity is evident in that the immune system produces far more antibodies than will be effectively used in binding with an antigen. In fact, it appears that the majority of antibodies produced do not play any active role whatsoever in the immune response. Natural variation is provided by the variable gene regions responsible for the production of highly diverse population of antibodies, and selection occurs, such that only antibodies able to successfully bind with an antigen will reproduce and be maintained as memory cells.

The similarity between adaptive biological evolution and the production of antibodies is even more striking when one considers that the two central processes involved in the production of antibodies, genetic recombination and mutation, are the same ones responsible for the biological evolution of sexually reproducing species. The recombination and editing of immunoglobulin genes underlies the large diversity of the antibody population, and the mutation of these genes serves as a fine-tuning mechanism (see Section II.B). In sexually reproducing species, the same two processes are involved in providing the variations on which natural selection can work to fit the organism to the environment [26]. Thus, cumulative blind variation and natural selection, which over many millions of years resulted in the emergence of mammalian species, remain crucial in the day-by-day ceaseless battle to survival of these species. It should, however, be noted that recombination of immunoglobulin genes involved in the production of antibodies differs somewhat from the recombination of parental genes in sexual reproduction. In the former, nucleotides can be inserted and deleted at random from recombined immunoglobulin gene segments and the latter involves the crossing-over of parental genetic material, generating an offspring that is a genetic mixture of the chromosomes of its parents.

Whereas adaptive biological evolution proceeds by cumulative natural selection *among* organisms, research on the immune system has now provided the first clear evidence that ontogenetic adaptive changes can be achieved by cumulative blind variation and selection *within* organisms [25]. The clonal selection algorithm (CLONALG), to be described further in the text, aims at demonstrating that this cumulative blind variation can generate high quality solutions to complex problems.

IV. THE SHAPE-SPACE MODEL

The shape-space model aims at quantitatively describing the interactions among antigens and antibodies (Ag-Ab) [27]. The set of features that characterize a molecule is called its *generalized shape*. The Ag-Ab representation (binary or real-valued) determines a distance measure to be used to calculate the degree of interaction between these molecules. Mathematically, the generalized shape of a molecule (m), either an antibody or an antigen, can be represented by a set of coordinates $m = \langle m_L, ..., m_2, m_1 \rangle$, which can be regarded as a point in an L-dimensional real-valued shape-space ($m \in S^L$). The precise physical meaning of each parameter is not relevant to the development of computational tools. In this work, we consider binary (or integer) strings to represent the molecules. Antigens and antibodies assume the same length L. The length and cell representation depends upon each problem, as will be seen in Section VII.

V. COMPUTATIONAL ASPECTS OF THE CLONAL SELECTION PRINCIPLE

After discussing the clonal selection principle and the affinity maturation process, the development and implementation of the clonal selection algorithm (CLONALG) is straightforward. CLONALG will be initially proposed to perform machine-learning and pattern recognition tasks, and then adapted to be applied to optimization problems. The main immune aspects taken into account to develop the algorithm were: maintenance of a specific memory set, selection and cloning of the most stimulated antibodies, death of non-stimulated antibodies, affinity maturation and re-selection of the clones proportionally to their antigenic affinity, generation and maintenance of diversity.

Consider the following notation, where bold face expressions (e.g. **Ag** and **Ab**) indicate matrices, bold face italic letters (e.g. Ag_j) indicate vectors, and S corresponds to a coordinate axis in the shape-space:

- **Ab**: available antibody repertoire ($\mathbf{Ab} \in S^{N \times L}$, $\mathbf{Ab} = \mathbf{Ab}_{\{r\}} \cup \mathbf{Ab}_{\{m\}}$);
- $\mathbf{Ab}_{\{m\}}$: memory antibody repertoire ($\mathbf{Ab}_{\{m\}} \in S^{m \times L}$, $m \leq N$);
- $\mathbf{Ab}_{\{r\}}$: remaining antibody repertoire ($\mathbf{Ab}_{\{r\}} \in S^{r \times L}$, $r = N - m$);
- $\mathbf{Ag}_{\{M\}}$: population of antigens to be recognized ($\mathbf{Ag}_{\{M\}} \in S^{M \times L}$);
- f_j: vector containing the affinity of all antibodies with relation to antigen Ag_j;
- $\mathbf{Ab}_{\{n\}}^{j}$: n antibodies from **Ab** with highest affinities to Ag_j ($\mathbf{Ab}_{\{n\}}^{j} \in S^{n \times L}$, $n \leq N$);
- $\mathbf{C}^{j}$: population of N_c clones generated from $\mathbf{Ab}_{\{n\}}^{j}$ ($\mathbf{C}^{j} \in S^{N_c \times L}$);
- $\mathbf{C}^{j*}$: population $\mathbf{C}^{j}$ after the affinity maturation (hypermutation) process;
- $\mathbf{Ab}_{\{d\}}$: set of d new molecules that will replace d low-affinity antibodies from $\mathbf{Ab}_{\{r\}}$ ($\mathbf{Ab}_{\{d\}} \in S^{d \times L}$, $d \leq r$);
- Ab_j^{*}: candidate, from $\mathbf{C}^{j*}$, to enter the pool of memory antibodies.

A. Pattern Recognition

In this case, an explicit antigen population **Ag** is available for recognition. Without loss of generality, it will be assumed the pre-existence of one memory antibody, from $\mathbf{Ab}_{\{m\}}$ to each antigen to be recognized. The CLONALG algorithm can be described as follows (see Fig. 4):

1. Randomly choose an antigen Ag_j ($Ag_j \in \mathbf{Ag}$) and present it to all antibodies in the repertoire $\mathbf{Ab} = \mathbf{Ab}_{\{r\}} \cup \mathbf{Ab}_{\{m\}}$ ($r + m = N$);
2. Determine the vector f_j that contains its affinity to all the N antibodies in **Ab**;

3. Select the *n* highest affinity antibodies from **Ab** composing a new set $\mathbf{Ab}_{\{n\}}^{j}$ of high affinity antibodies with relation to *Ag$_j$*;

4. The *n* selected antibodies will be cloned (reproduced) independently and proportionally to their antigenic affinities, generating a repertoire $\mathbf{C}^{j}$ of clones: the higher the antigenic affinity, the higher the number of clones generated for each of the *n* selected antibodies;

5. The repertoire $\mathbf{C}^{j}$ is submitted to an affinity maturation process inversely proportional to the antigenic affinity, generating a population $\mathbf{C}^{j*}$ of matured clones: the higher the affinity, the smaller the mutation rate;

6. Determine the affinity f_j^* of the matured clones $\mathbf{C}^{j*}$ with relation to antigen *Ag$_j$*;

7. From this set of mature clones $\mathbf{C}^{j*}$, re-select the highest affinity one (Ab_j^*) with relation to *Ag$_j$* to be a candidate to enter the set of memory antibodies $\mathbf{Ab}_{\{m\}}$. If the antigenic affinity of this antibody with relation to *Ag$_j$* is larger than its respective memory antibody, then Ab_j^* will replace this memory antibody; and

8. Finally, replace the *d* lowest affinity antibodies from $\mathbf{Ab}_{\{r\}}$, with relation to *Ag$_j$*, by new individuals.

After presenting all the *M* antigens from **Ag** and performing the eight steps above to all of them, we say a *generation* has been completed. Appendix I presents the pseudocode to implement this version of the CLONALG algorithm.

In our implementation, it was assumed that the *n* highest affinity antibodies were sorted in ascending order after Step 3, so that the amount of clones generated for all these *n* selected antibodies was given by (1):

$$N_c = \sum_{i=1}^{n} round\left(\frac{\beta . N}{i}\right),$$ (1)

where N_c is the total amount of clones generated for each of the antigens, β is a multiplying factor, *N* is the total amount of antibodies and *round*(·) is the operator that rounds its argument towards the closest integer. Each term of this sum corresponds to the clone size of each selected antibody, e.g., for *N* = 100 and β = 1, the highest affinity antibody (*i* = 1) will produce 100 clones, while the second highest affinity antibody produces 50 clones, and so on.

B. *Optimization*

To apply the proposed CLONALG algorithm to optimization tasks, a few modifications have to be made in the algorithm as follows:

- In Step 1, there is no explicit antigen population to be recognized, but an objective function $g(\cdot)$ to be optimized (maximized or minimized). This way, an antibody affinity corresponds to the evaluation of the objective function for the given antibody: each antibody Ab_i represents an element of the input space. In addition, as there is no specific antigen population to be recognized, the whole antibody population **Ab** will compose the memory set and, hence, it is no longer necessary to maintain a separate memory set $Ab_{\{m\}}$; and

- In Step 7, n antibodies are selected to compose the set **Ab**, instead of selecting the single best individual Ab^*.

If the optimization process aims at locating multiple optima within a single population of antibodies, then two parameters may assume default values:

1. Assign $n = N$, i.e. all antibodies from the population **Ab** will be selected for cloning in Step 3; and

2. The affinity proportionate cloning is not necessarily applicable, meaning that the number of clones generated for each of the N antibodies is assumed to be the same, and (1) becomes

$$N_c = \sum_{i=1}^{N} round(\beta.N), \qquad (2)$$

The second statement above, has the implication that each antibody will be viewed locally and have the same clone size as the other ones, not privileging anyone for their affinity. The antigenic affinity (corresponding to the objective function value) will only be accounted to determine the hypermutation rate for each antibody, which is still proportional to their affinity.

Note: In order to maintain the best antibodies for each clone during evolution, it is possible to keep one original (parent) antibody for each clone unmutated during the maturation phase (Step 5).

Fig. 5 presents the block diagram to implement the clonal selection algorithm (CLONALG) adapted to be applied to optimization tasks. The pseudocode for this algorithm is described in Appendix II. Notice that there are slight differences compared to the previous version depicted in Fig. 4.

VI. EVOLUTIONARY ALGORITHMS AND NICHING METHODS

As will be seen in the discussion (Section IX), CLONALG has many properties in common with well-known *evolutionary algorithms* (EAs), and can be applied to the same settings. Due to these similarities, a brief exposition about genetic algorithms and niching methods will be presented here.

Genetic algorithms (GAs) constitute stochastic evolutionary techniques whose search methods model some natural phenomena: genetic inheritance and Darwinian strife for survival [26]. GAs perform a search through a space of potential solutions, and the space is characterized by the definition of an evaluation (fitness) function, which plays the role of an environmental feedback. A genetic algorithm (or any evolutionary algorithm), must have the following five components [28]:

1. A genetic representation for potential candidate solutions;
2. A way to create an initial population of potential solutions;
3. An evaluation (*fitness*) function;
4. Genetic operators to define the composition of an offspring; and
5. Values for the various parameters used by the algorithm: population size, genetic operator rate, etc.

Fig. 6 depicts the block diagram for a standard evolutionary algorithm. In typical applications of genetic algorithms, each generation produces increasingly fit organisms among an explosive number of candidates in highly uncertain environments. The algorithm is used to evolve a population of potential candidate solutions, where a single member is going to specify the best solution, i.e., a fitness peak. Particularly in the case of GAs, the selection and genetic operators (crossover, mutation and inversion) guide the population towards its fittest members.

Niching methods extend genetic algorithms to domains that require the location and maintenance of multiple solutions, like multimodal and multiobjective function optimization [29], [30]. Niching methods can be divided into families or categories, based upon structure and behavior. To date, two of the most successful categories of niching methods are *fitness sharing* and *crowding*. Both categories contain methods that are capable of locating and maintaining multiple solutions within a population, whether these solutions have identical or distinct values (fitnesses).

Fitness sharing, first introduced by Goldberg and Richardson [31], is a fitness scaling mechanism that alters only the fitness assignment stage of a GA. From a multimodal function optimization perspective, the idea behind sharing is as follows. If similar individuals are required to share fitness, then the number of individuals that can reside in any portion of the fitness landscape is limited by the

fitness of that portion of the landscape. Sharing results in individuals being allocated to optimal regions of the fitness landscape. The number of individuals residing near any peak will theoretically be proportional to the height of that peak. Sharing works by derating each population element's fitness by an amount related to the number of similar individuals in the population. Specifically an individual shared fitness f_s is given by:

$$f_s(x_i) = \frac{f(x_i)}{\sum_{j=1}^{N} sh(d(x_i, x_j))},$$

(3)

where $sh(\cdot)$ is the sharing function given by (4) :

$$sh(d) = \begin{cases} 1 - \left(\dfrac{d}{\sigma_{share}}\right)^{\alpha}, & \text{if } d < \sigma_{share}, \\ 0, & \text{otherwise} \end{cases}$$

(4)

where α is a constant that regulates the shape of the sharing function (usually set to 1), and σ_{share} represents a threshold of dissimilarity. A sharing GA, GA_{sh} for short, can distinguish its niches by employing a *genotypic* or *phenotypic* distance measure.

VII. PERFORMANCE EVALUATION

In order to evaluate the CLONALG capability to perform machine-learning and optimization, it was applied it to several benchmark problems:

1. A binary character recognition task was used to test its learning and memory acquisition capabilities;
2. Three function optimization tasks evaluated its potential to perform multimodal search; and
3. A 30-city instance of the Travelling Salesman Problem (TSP) was used to test its efficacy in solving combinatorial optimization problems.

In all runs of the algorithm, the stopping criterion was a pre-defined maximum number of generations (N_{gen}).

A. Pattern Recognition

The learning and memory acquisition of the CLONALG is investigated through its application to a binary character recognition problem. The goal is to demonstrate that a cumulative blind variation together with selection can produce individuals with increasing affinities (maturation of the immune response). In this case, it is assumed that the antigen population to be learnt and recognized is

represented by a set of eight binary characters. These characters are the same ones originally proposed by Lippmann [32], in a different context. Each character is represented by a bitstring (Hamming shape-space, briefly discussed in Section IV) of length $L = 120$ (the resolution of each picture is 12×10). The antibody repertoire is composed of $N = 10$ individuals, where $m = 8$ of them belong to the memory set $\mathbf{Ab}_{\{m\}}$. The other running parameters were $N_{gen} = 50$, $n = 5$, $d = 0.0$ and $\beta = 10$.

The original characters (antigens) are depicted in Fig. 7(a). Fig. 7(b) illustrates the initial memory set, and Figs. 7(c) to 7(e) represent the maturation of the memory set (immune response) through generations. The affinity measure takes into account the Hamming distance (D) between antigens and antibodies, according to (5):

$$D = \sum_{i=1}^{L} \delta, \text{where} \quad \delta = \begin{cases} 1 & \text{if } Ab_i \neq Ag_i \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

The algorithm converged after 250 cell generations.

Notice that an exact matching is not necessary to obtain a successful character recognition. A partial matching is enough in most applications. This would allow us to define an affinity threshold ε, that could be decisive to determine the final size and configuration of the memory set.

B. Multimodal Optimization

The CLONALG reproduces those individuals with higher affinities and selects their improved maturated progenies. This strategy suggests that the algorithm performs a greedy search, where single members will be locally optimized (exploitation of the surrounding space), and the newcomers yield a broader exploration of the search-space as proposed by the receptor editing process described in Section II.B. This characteristic makes the CLONALG very suitable for solving multimodal optimization tasks.

Consider the cases in which we intend to maximize the following functions:

1. $g_1(x) = \sin^6(5\pi x)$;

2. $g_2(x) = 2^{(-2((x-0.1)/0.9)^2)} \sin^6(5\pi x)$; and

3. $g_3(x,y) = x.\sin(4\pi x) - y.\sin(4\pi y + \pi) + 1$.

The first two unidimensional functions were used by Bäck *et al.* [29], Goldberg and Richardson [31] and Goldberg [33] to evaluate *niching methods* for genetic algorithms. A discussion relating CLONALG with evolutionary algorithms, including fitness sharing methods, will be presented in Section IX. In this section, the CLONALG and GA_{sh} algorithms will be applied to optimize the three

functions above and comparisons will be done based upon the quality of the solutions obtained. The GA$_{sh}$ was implemented using a binary tournament selection with $k = 0.6$ [34], crossover probability $p_c = 0.6$, mutation probability $p_m = 0.0$ and $\sigma_{share} = 4$, in all cases. To promote stability at the combination of sharing and tournament selection, we used the *continuous updating method*, as proposed by Oei *et al.* [35].

In all cases, we employed the Hamming shape-space, with binary strings representing coded real values for the input variables of the functions $g_i(\cdot)$, $i = 1,...,3$. The chosen bitstring length was $L = 22$, corresponding to a precision of six decimal places. The mapping from a binary string $m = \langle m_L,..., m_2, m_1 \rangle$ into a real number z is completed in two steps:

1. Convert the binary string $m = \langle m_L,..., m_2, m_1 \rangle$ from base 2 to base 10:
    $$\left(\langle m_L,...,m_2,m_1 \rangle \right)_2 = \left(\sum_{i=0}^{21} m_i.2^i \right)_{10} = z'\text{; and}$$

2. Find the corresponding real value for z: $z = z_{min} + z'.\dfrac{z_{max} - z_{min}}{2^{22} - 1}$, where z_{max} is the upper bound of the interval in which the variable is defined, and z_{min} is the lower bound of this interval.

For functions g_1 and g_2, variable x is defined over the range [0,1], ($z_{min} = 0$, $z_{max} = 1$), and in the case of function g_3, the variables x and y are defined over the range [−1,2], ($z_{min} = −1$, $z_{max} = 2$). The affinity measure corresponds to the evaluation of the respective functions after decoding x and y (if applicable), as described above. Fig. 8 presents the results produced by CLONALG after 50 generations. Notice that the solutions (circles) cover all the peaks in both cases.

The running parameters adopted were

- Functions $g_1(x)$ and $g_2(x)$: $N_{gen} = 50$, $n = N = 50$, $d = 0.0$ and $\beta = 0.1$ according to (2); and
- Function $g_3(x,y)$: $N_{gen} = 50$, $n = N = 100$, $d = 0.0$ and $\beta = 0.1$ according to (2).

Fig. 9 presents the performance of the fitness sharing technique (GA$_{sh}$) discussed in Section VI, when applied to the same problems, and Fig. 10 presents the performance of CLONALG and GA$_{sh}$ when applied to the problem of maximizing function $g_3(x,y)$. Section IX will bring general discussions concerning the performance of the algorithms with relation to all the problems tested.

C. Combinatorial Optimization

As a last evaluation for the CLONALG algorithm, it was applied to a 30-city instance of the *travelling salesman problem* (TSP). Simply stated, the travelling salesman must visit every city in his

territory, exactly once, and then return to the starting city. The question is: given the cost of travel between all pairs of cities, which is the tour with the smallest cost? In this work, the cost of the tour is basically the length of the itinerary traveled by the salesman.

The TSP is a typical case of a combinatorial optimization problem and arises in numerous applications, from VLSI circuit design, to fast food delivery. In this case, the use of an Integer shape-space might be appropriate, where integer-valued vectors of length L, composed of permutations of elements in the set $C = \{1,2,...,L\}$, represent the possible tours. Each component of the integer vector indexes a city. The total length of each tour gives the affinity measure of the corresponding vector. Mutation is simply performed by swapping pairs of cities in the antibodies that represent the tours.

Figure 11 presents the best solution determined by the CLONALG algorithm, which corresponds to the global optimum [36], after 300 generations. The population size is $N = 300$ antibodies, with $d = 60$ (corresponding to a rate of 20% newcomers). In this case, low affinity individuals are allowed to be replaced after each 20 generations. This scheduling is supposed to leave a breathing time in order to allow the achievement of local optima, followed by the replacement of the poorer individuals. The other parameters adopted by the CLONALG algorithm were $N_{gen} = 300$, $n = 150$ and $\beta = 2$, according to (1).

VIII. SENSITIVITY ANALYSIS

The proposed clonal selection algorithm has deterministic selection procedures and performs only mutation (there is no crossover), hence it is not necessary to define a crossover probability p_c, as usual in evolutionary algorithms. However, CLONALG has three user-defined parameters that influence mainly 1) the convergence speed, 2) the computational complexity (see Appendix III) and 3) its capability to perform a multimodal search:

- n (Steps 3 and 7): number of antibodies to be selected for cloning, giving rise to population $\mathbf{Ab}_{\{n\}}$;
- N_c (Step 4): number of clones generated from $\mathbf{Ab}_{\{n\}}$ (proportional to β according to (1) or (2)); and
- d (Step 8): amount of low affinity antibodies to be replaced.

In order to study the effects of setting these parameters when running the CLONALG algorithm, consider the problem of maximizing function $g_1(x) = \mathrm{sen}^6(5\pi x)$ as described in Section VII.B. The antibody population will be assumed of fixed size $N = 10$ (twice the number of peaks).

A. Sensitivity With Relation to Parameter n

To evaluate the CLONALG sensitivity with relation to n, we fixed $\beta = 1$, resulting in $N_c = 100$ according to (2), and $d = 0.0$. Parameter n is going to take the values $n = \{2,4,6,8,10\}$. For $n < N$, the algorithm presents difficulties in locating all the optima for the given function, then it will be assumed to have converged when at least one of the 5 peaks is located. The number of generations N_{gen} necessary for convergence was evaluated, as described in Fig. 12(a). The results presented are the maximum, minimum and mean obtained over 10 runs of the algorithm for each value of n. It can be noticed from this picture that the parameter n does not strongly influence the number of generations required to locate one maximum of the function. On the other side, n has a strong influence on the size of the population $\mathbf{Ab}_{\{n\}}$, as described by (2), and larger values of n imply in a higher computational cost to run the algorithm (see Appendix III, Table I).

Fig. 13(b) shows that the average population affinity rises as n increases. This behavior was expected in this particular case, because the higher the value of n, the larger the number of antibodies that will be located at a maximum.

B. Sensitivity With Relation to Parameter N_c

In order to study the CLONALG sensitivity with relation to N_c, n was set to $n = N = 10$ (default for multimodal optimization), $d = 0.0$, and N_c assumed the following values: $N_c = \{5,10,20,40,80,160\}$. In this case, the algorithm is supposed to have converged after locating all the five peaks of the function and the average fitness, f_{av}, of the whole antibody population $\mathbf{Ab}$ was larger than $f_{av} = 0.92$, where the maximum average value is $f_{av} = 1.0$. Fig. 14 shows the trade-off between N_c and the number of generations necessary for convergence. The results presented are the maximum, minimum and mean obtained over 10 runs of the algorithm for each value of N_c.

It can be seen from the results presented in this figure that the higher N_c (or β), the faster the convergence, in terms of number of generations. Nevertheless, the coputational time per generation increases linearly with N_c, as discussed in Appendix III.

C. Sensitivity With Relation to Parameter d

The number d of low affinity antibodies to be replaced (Step 8 of the algorithm) is of extreme importance for the introduction and maintenance of diversity in the population, and thus for the CLONALG potentiality of exploring new regions of the affinity landscape, as discussed in Section

II.B and illustrated in Fig. 3. This was verified through the generation of an initial population composed of $N = 10$ identical antibodies, $\mathbf{Ab}_i = \langle 0,...,0 \rangle$, $i = 1,...,10$. The initial affinity of all these antibodies is $f_i = 0$, $i = 1,...,10$, and the default value for multimodal optimization is $n = N$. The algorithm was run for $d = 0.0$, $d = 1$ and $d = 2$, corresponding to 0%, 10% and 20% of the population being randomly replaced at each generation. Fig. 14 shows the CLONALG capability of locating all optima for $d = 0.0$ and $d = 2$. It can be noticed that for values of $d > 0$, the algorithm can locate all the peaks of the function to be maximized. Nevertheless, it is important to note that very high values for this parameter may result in a random search through the affinity landscape, being suggested values ranging from 5% to 20% of N (size of the antibody repertoire).

IX. Discussion

A. Theoretical Aspects

The CLONALG algorithm, as proposed in this paper, represents a computational implementation of the clonal selection principle responsible for describing the behavior of the B cells during an adaptive immune response (Section II). In both implementations, Figs. 4 and 5, it is assumed the existence of an antibody repertoire (**Ab**) that will be stimulated by an antigen (that can be an explicit population **Ag** to be recognized, or a value of an objective function $g(\cdot)$ to be optimized), and those higher affinities antibodies will be selected to generate a population of clones. During the cloning process, mitosis, a few antibodies will suffer somatic mutation proportional to their antigenic affinities, and the highest affinity clones will be selected to compose a memory set. Low affinity antibodies are replaced, simulating the process of receptor editing (and/or apoptosis followed by a random reshuffling of the antibody genes in order to generate a new receptor).

By comparing CLONALG (Figs. 4 and 5) with the evolutionary algorithms described in Section VI, Fig. 6, it is noticeable that the main steps composing the EAs are embodied in CLONALG, allowing us to characterize it as an evolutionary-like algorithm. However, while evolutionary algorithms use a vocabulary borrowed from natural genetics and are inspired in the Darwinian evolution, the proposed clonal selection algorithm (CLONALG) makes use of the shape-space formalism, along with immunological terminology to describe Ag-Ab interactions and cellular evolution, as discussed in Section III. The CLONALG performs its search through the mechanisms of somatic mutation and receptor editing, balancing the exploitation of the best solutions with the exploration of the search-space. Essentially, its encoding scheme is not different from that of genetic algorithms.

The proposed clonal selection algorithm can also be characterized as a cooperative and competitive approach, where individual antibodies are competing for antigenic recognition (or optimization) but the whole population (at least the specific memory set) will cooperate as an ensemble of individuals to present the final solution.

Similarly to CLONALG, the *evolution strategies* (ESs) [37], [38] also employ self-adapting genetic variation mechanisms. The ESs consider the mutation parameter as the main genetic variation operator, and recombination is mainly required to control mutation. They use gaussian-like mutations and theoretical rules to control its rate [39]. On the other side, the approach proposed for the CLONALG algorithm is plausible from the biological standpoint and can be implemented through rules like the one described in (6) and illustrated in Fig. 15:

$$\alpha = \exp(-\rho.f), \tag{6}$$

where α is the mutation rate, ρ controls its decay and f is the antigenic affinity. In this function, the antigenic affinity and mutation rates are normalized over the [0,1] interval.

What should be emphasized, however, is that the genetic variation rates are proportional to the individuals' affinities in both cases.

B. Empirical Aspects

The results presented in Sections VII and VIII allow us to infer several important properties of the proposed algorithm.

First, and most important, while adaptive biological evolution occurs through natural selection *among* organisms (or among organisms and selfish genes [40]), research in immunology has now provided the first clear evidence that ontogenetic adaptive changes can be achieved through variation and selection *within* organisms (see discussion in Section III).

By comparing the performance of the proposed CLONALG algorithm for multimodal optimization with the fitness sharing method GA_{sh} briefly presented in Section VI, we can note that the number of nonpeak individuals located by the CLONALG algorithm is much smaller for functions $g_1(x)$ and $g_2(x)$ (Fig. 8) than those determined by the GA_{sh} (Fig. 9). In the case of function $g_3(x)$, where the separation among peaks is non-uniform, the fitness sharing overlooked peaks distributed with less uniformity and privileged the highest ones, as illustrated in Fig. 10, while CLONALG located a higher amount of local optima, including the global maximum of the function.

While GA_{sh} has to set parameter σ_{share}, what is a difficult task that might depend upon the number of peaks (usually unknown a priori), and recombination probabilities p_c ($p_m = 0.0$ in most cases), CLONALG has parameters n, d and β (or N_c) to be defined (n is usually equal to N for multimodal search). The sensitivity analysis of the algorithm (Section VIII) demonstrated that n is crucial to the algorithms' capability of locating a large number of local optima, d influences the CLONALG potential to explore new areas of the affinity landscape and β is strongly related to the convergence speed and computational time required to run the algorithm. As a final aspect of the GA_{sh}/CLONALG comparison, the GA_{sh} has an associated computational cost of order $O(N^2)$ (every population member is compared to all others), while CLONALG has an associated computational cost of order $O(N_c L)$ for multimodal function optimization, as detailed in Appendix III. Each antibody optimized by CLONALG performs a sort of hill climbing from its initial position, with the possibility of moving from a lower to a higher peak, depending upon its mutation rate. This way, unlike fitness sharing, the number of individuals around each peak is not theoretically proportional to its fitness (affinity) value, but to its initial configuration.

X. CONCLUDING REMARKS

In this paper, we presented a general-purpose algorithm inspired in the clonal selection principle and affinity maturation process of the adaptive immune response. The algorithm was verified to be capable of performing learning and maintenance of high quality memory and, it was also capable of solving complex engineering tasks, like multimodal and combinatorial optimization.

Two versions of the algorithm, named CLONALG, were derived. It was initially proposed to perform machine-learning and pattern recognition tasks, where an explicit antigen population, representing a set of input patterns, was available for recognition. Due to its evolutionary characteristics, the algorithm was adapted to solve optimization problems, emphasizing the multimodal case. The values provided by the function being optimized for a given input may be associated with the affinity to an antigen.

No distinction between the cell and its receptor is made, thus the editing and cell replacement processes are equivalent, leading to diversity introduction and broader exploration of the affinity landscape. In addition, the proposed CLONALG algorithm constitutes a crude version of the clonal selection principle. Many heuristics, e.g. a specific type of mutation, may be inserted in order to improve its performance in solving particular tasks, like the travelling salesman problem exemplified in Section VII.C.

By comparing the proposed algorithm with the standard genetic algorithm (GA), we can notice that the CLONALG can reach a diverse set of local optima solutions, while the GA tends to polarize the whole population of individuals towards the best candidate solution. Essentially, their coding schemes and evaluation functions are not different, but their evolutionary search processes differ from the viewpoint of inspiration, vocabulary and sequence of steps, though CLONALG embodies the 5 main steps required to characterize an evolutionary algorithm. In the particular case of multimodal optimization, it emerges the possibility of applying the algorithm to problems in classification, multiobjective function optimization and simulation of complex and adaptive systems [29].

Another important aspect of our model, when compared with the standard GA, is the fact that the CLONALG takes into account the cell affinity, corresponding to an individual's fitness, in order to define the mutation rate to be applied to each member of the population. The GA usually adopts rates for the genetic operators that disregard the individual fitness. Evolution strategies typically adopt self-adapting parameters, but under a mathematical framework completely different from the biologically inspired mechanism used by CLONALG.

Acknowledgement

REFERENCES

[1] Y. Ishida, "The Immune System as a Self-Identification Process: A Survey and a Proposal", in *Proc. of the IMBS'96*, 1996.

[2] J. E. Hunt and D. E. Cooke, "Learning Using an Artificial Immune System", *Journal of Network and Computer Applications*, vol. 19, pp. 189-212, 1996.

[3] D. Dasgupta, *Artificial Immune Systems and Their Applications*, Ed., Springer-Verlag, 1999.

[4] S. A. Hofmeyr and S. Forrest, "Immunity by Design: An Artificial Immune System", in *Proc. of GECCO'99*, 1999, pp. 1289-1296.

[5] L. N. de Castro and F. J. Von Zuben, "Artificial Immune Systems: Part II – A Survey of Applications", Tech. Rep. – RT DCA 02/00, p. 65. [Online]. Available: http://www.dca.fee.unicamp.br/~lnunes/immune.html

[6] ——, "Artificial Immune Systems: Part I – Basic Theory and Applications", FEEC/Unicamp, Campinas, SP, Tech. Rep. – RT DCA 01/99, p. 95. [Online]. Available: http://www.dca.fee.unicamp.br/~lnunes/immune.html

[7] F. M. Burnet, "Clonal Selection and After", in *Theoretical Immunology*, G. I. Bell, A. S. Perelson and G. H. Pimbley Jr., Eds., Marcel Dekker Inc., 1978, pp. 63-85.

[8] ——, *The Clonal Selection Theory of Acquired Immunity*, Cambridge University Press, 1959.

[9] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, A Bradford Book, 1998.

[10] C. A. Janeway, P. Travers, M. Walport, and J. D. Capra, *Immunobiology The Immune System in Health and Disease*, Artes Médicas (in Portuguese), 4th Ed, 2000.

[11] D. J. Smith, S. Forrest, R. R. Hightower and S. A. Perelson, "Deriving Shape Space Parameters from Immunological Data", *J. Theor. Biol.*, vol. 189, pp. 141-150, 1997.

[12] P. D. Hodgkin, "Role of Cross-Reactivity in the Development of Antibody Responses", *The Immunologist*, vol. 6, no. 6, pp. 223-226, 1998.

[13] G. L. Ada and G. Nossal, "The Clonal Selection Theory", *Scientific American*, vol. 257, no. 2, pp. 50-57, 1987.

[14] J. Sprent, "T and B Memory Cells", *Cell*, vol. 76, pp. 315-322, 1994.

[15] D. Mason, "Antigen Cross-Reactivity: Essential in the Function of TCRs", *The Immunologist*, vol. 6, no. 6, pp. 220-222, 1998.

[16] S. Haykin, *Neural Networks – A Comprehensive Foundation*, Prentice Hall, 2nd Ed., 1999.

[17] D. Allen, A. Cumano, R. Dildrop, C. Kocks, K. Rajewsky, N. Rajewsky, J. Roes, F. Sablitzky and M. Siekevitz, "Timing, Genetic Requirements and Functional Consequences of Somatic Hypermutation During B-Cell Development", *Imm. Rev.*, vol. 96, pp. 5-22, 1987.

[18] A. Coutinho (1989), "Beyond Clonal Selection and Network", *Immun. Rev.*, vol. 110, pp. 63-87, 1989.

[19] R. Ahmed and J. Sprent, "Immunological Memory", *The Immunologist*, vol. **7**, nos. 1-2, pp. 23-26, 1999.

[20] C. Berek and M. Ziegner, "The Maturation of the Immune Response", *Imm. Today*, vol. 14, no. 8, pp. 400-402, 1993.

[21] A. J. T. George and D. Gray, "Receptor Editing During Affinity Maturation", *Imm. Today*, vol. 20, no. 4, pp. 196, 1999.

[22] M. C. Nussenzweig, "Immune Receptor Editing; Revise and Select", *Cell*, vol. 95, pp. 875-878, 1998.

[23] S. Tonegawa "Somatic Generation of Antibody Diversity", *Nature*, vol. 302, pp. 575-581, 1983.

[24] N. K. Jerne, "Idiotypic Networks and Other Preconceived Ideas", *Imm. Rev.*, 79, pp. 5-24, 1984.

[25] G. Cziko, "The Immune System: Selection by the Enemy", in *Without Miracles*, G. Cziko, A Bradford Book, The MIT Press, 1995, pp. 39-48.

[26] J. H. Holland, *Adaptation in Natural and Artificial Systems*, MIT Press, 5th ed. 1998.

[27] A. S. Perelson and G. F. Oster, "Theoretical Studies of Clonal Selection: Minimal Antibody Repertoire Size and Reliability of Self-Nonself Discrimination", *J. theor.Biol.*, vol. 81, pp. 645-670, 1979.

[28] T. Bäck, D. B. Fogel and Z. Michalewicz, *Evolutionary Computation 1 Basic Algorithms and Operators*, Institute of Physiscs Publishing (IOP), Bristol and Philadelphia, 2000a.

[29] ——, *Evolutionary Computation 2 Advanced Algorithms and Operators*, Institute of Physiscs Publishing (IOP), Bristol and Philadelphia, 2000b.

[30] S. W. Mahfoud, "Niching Methods for Genetic Algorithms", Illinois Genetic Algorithms Laboratory, Illinois, IL, IlliGAL Report no. 95001, May 1995.

[31] D. E. Goldberg and J. Richardson, "Genetic Algorithms With Sharing for Multimodal Function Optimization", in *Genetic Algorithms and Their Applications: Proc. of the Second International Conference on Genetic Algorithms*, 1987, pp. 41-49.

[32] R. P. Lippmann, "An Introduction to Computing with Neural Nets", *IEEE ASSP Magazine*, pp. 4-22, 1987.

[33] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addisson-Wesley Reading, Massachussets, 1989.

[34] M. Mitchel, *An Introduction to Genetic Algorithms*, The MIT Press, 1998.

[35] C. K. Oei, D. E. Goldberg and S.-J. Chang, "Tournament Selection, Niching and the Preservation of Diversity", Illinois Genetic Algorithms Laboratory, Illinois, IL, IlliGAL Report no. 91011, December 1991.

[36] P. Moscato and J. F. Fontanari, "Stochastic Versus Deterministic Update in Simulated Annealing", *Physics Letters A*, vol. 146, no. 4, pp. 204-208, 1990.

[37] I. Rechenberg, *Evolutionsstrategie: Optimierung Technischer Systeme Nach Prinzipien der Biologischen Evolution*, Frommann-Holzboog, Sttutgart, 1993.

[38] ——, "Evolution Strategy", in *Computational Intelligence Imitating Life*, J. M. Zurada, R. J. Marks II and C. J. Robinson, Eds. IEEE Press, 1994, pp. 147-159.

[39] T. Bäck and H. −P. Schwefel, "An Overview of Evolutionary Algorithms for Parameter Optimization", *Evolutionary Computation*, vol. 1, no. 1, pp. 1-23, 1993.

[40] R. Dawkins, *The Selfish Genes*, Oxford University Press, 1989.

[41] R. Johnsonbaugh, *Discrete Mathematics*, Prentice Hall, 4th ed., 1997.

[42] E. Horowitz and S. Sahni, *Fundamentals of Computer Algorithms*, London, Pitman, 1978.

[43] M. Fischetti and S. Martello, "A Hybrid Algorithm for Finding the kth Smallest of n Elements in $O(n)$ time", *Ann. Operations Res.*, vol. 13, pp. 401-419, 1988.
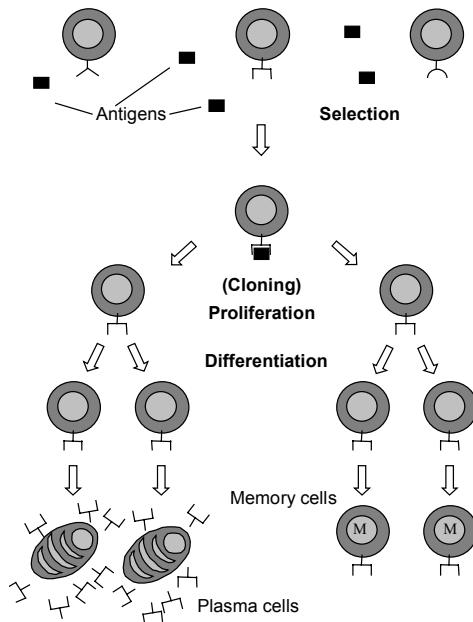
Fig. 1: The clonal selection principle.
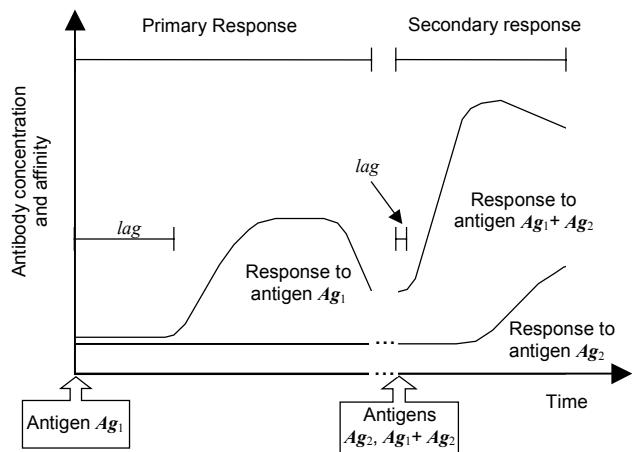
Fig. 2: Primary, secondary and cross-reactive immune responses. After an antigen has been seen once (*primary response*), subsequent encounters with the same antigen, or a related one (cross-reaction), will lead to a faster and stronger response (*secondary response*).
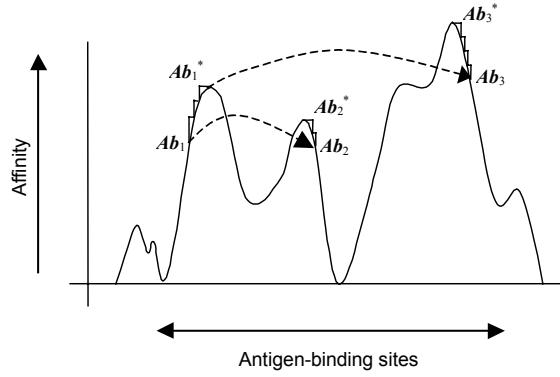
Fig. 3: Schematic representation of shape-space for antigen-binding sites. Somatic mutations guide to local optima, while receptor editing introduces diversity, leading to possibly better candidate receptors.
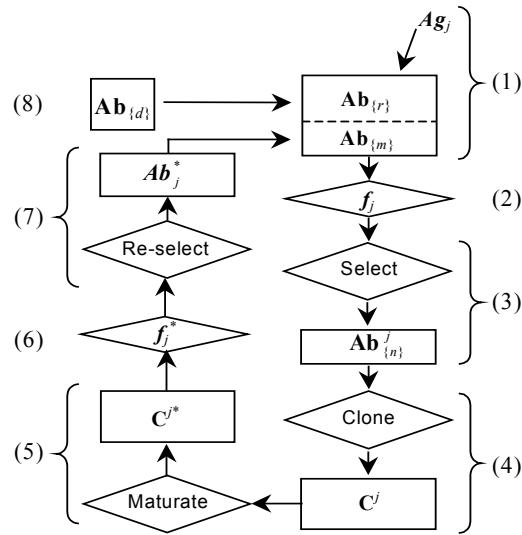

Fig. 4: Computational procedure for the clonal selection algorithm (CLONALG): pattern recognition case.
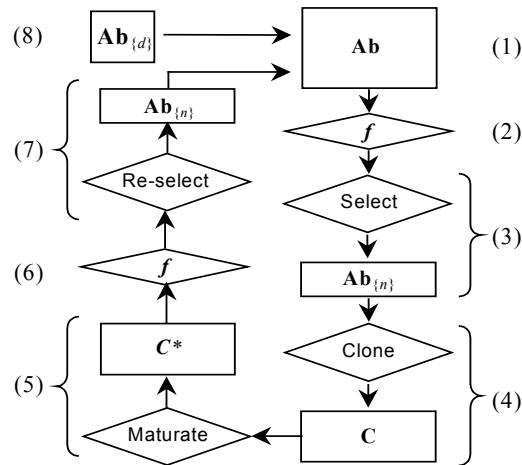
Fig. 5: Computational procedure for the clonal selection algorithm (CLONALG): optimization case.
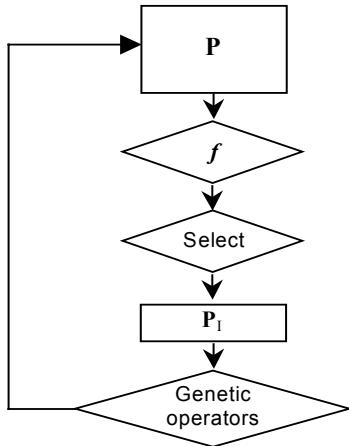
Fig. 6: Computational procedure for a standard evolutionary algorithm, where **P** represents the population at each generation, f is the vector containing the fitness values of all members of the population, and $\mathbf{P}_I$ is an intermediate population.

(a) Input patterns

(b) 0 generations

(c) 50 generations

(d) 100 generations
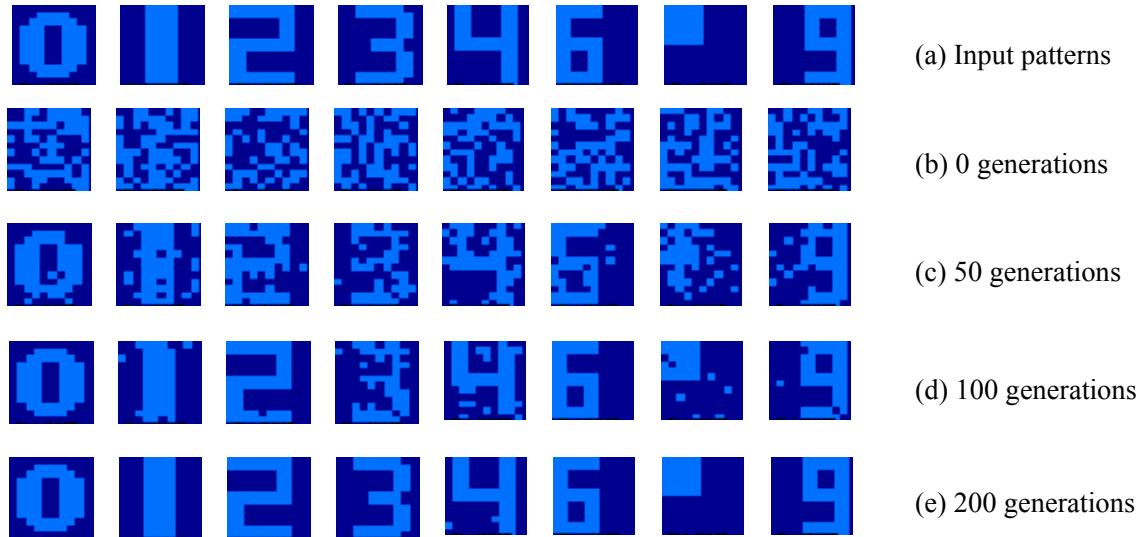
(e) 200 generations

Fig. 7: CLONALG applied to a pattern recognition problem. (a) Patterns to be learned, or input patterns (antigens). (b) Initial memory set. (c) Memory set after 50 cell generations. (d) Memory set after 100 cell generations. (e) Memory set after 200 cell generations.
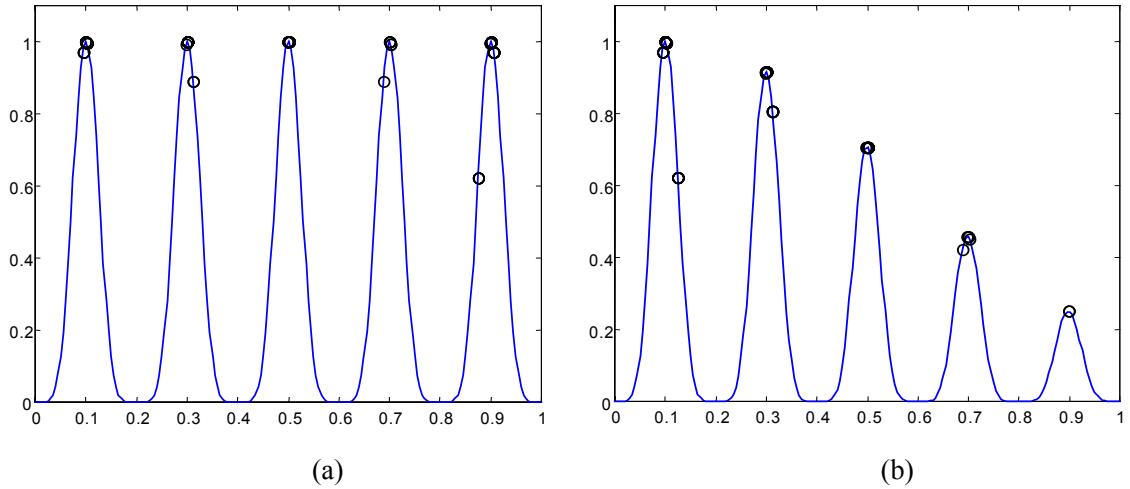
Fig. 8: CLONALG algorithm applied to the problem of maximizing functions $g_1(x)$ and $g_2(x)$. (a) $g_1(x) = \text{sen}^6(5\pi x)$. (b) $g_2(x) = 2^{(-2((x-0.1)/0.9)^2)}\text{sen}^6(5\pi x)$.
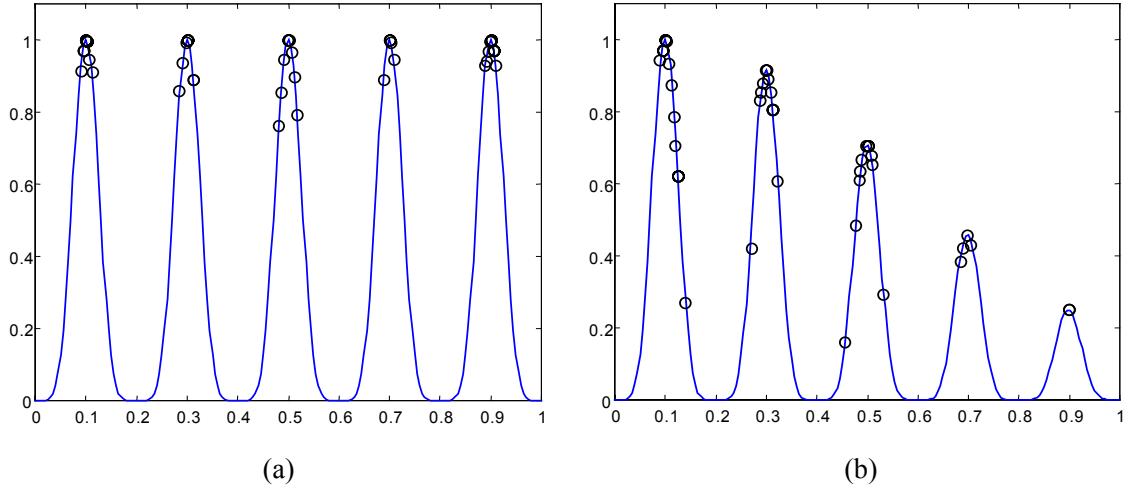

Fig. 9: GA$_{sh}$ applied to the problem of maximizing functions $g_1(x)$ and $g_2(x)$, $\sigma_{share} = 4$. (a) $g_1(x) = \text{sen}^6(5\pi x)$. (b) $g_2(x) = 2^{(-2((x-0.1)/0.9)^2)}\text{sen}^6(5\pi x)$.
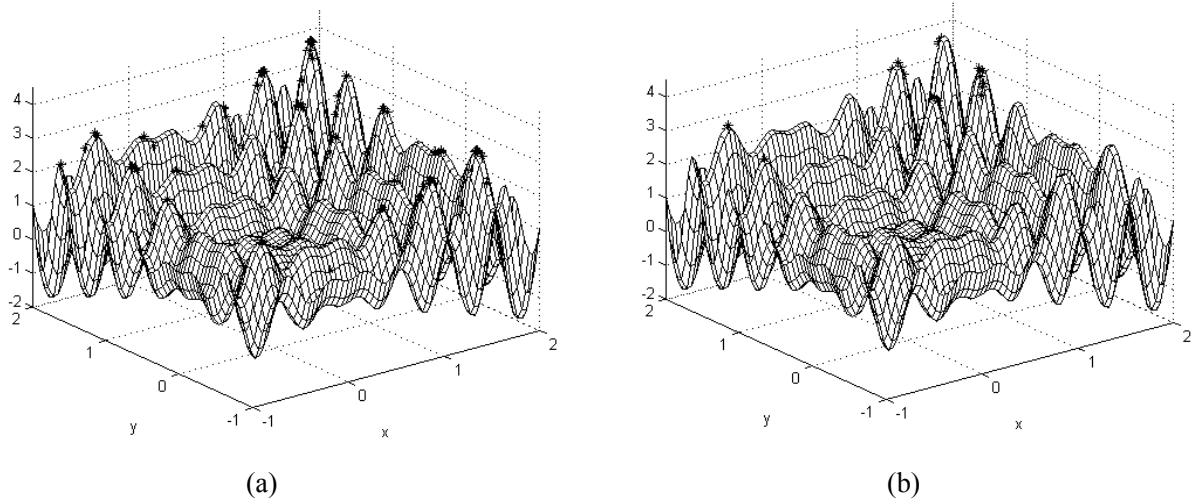

Fig. 10: Maximizing function $g_3(x,y) = x.\sin(4\pi x) - y.\sin(4\pi y + \pi) + 1$. (a) CLONALG. (b) GA$_{sh}$.
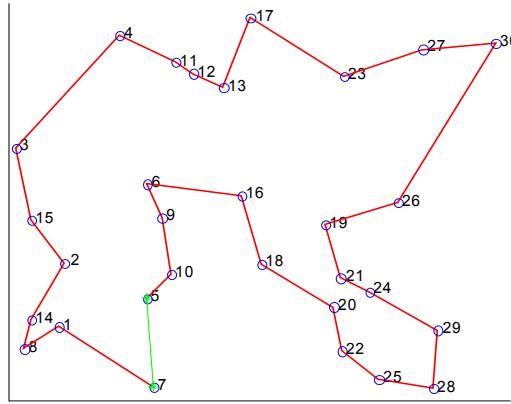
Fig. 11: Best tour determined by the CLONALG algorithm after 300 generations.


(a)

(b)
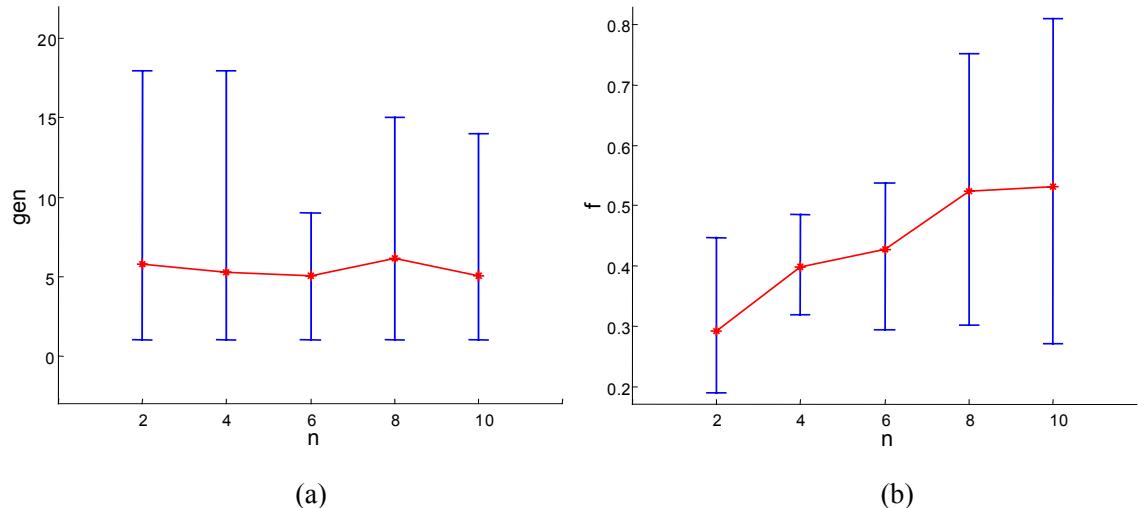
Fig. 12: CLONALG sensitivity with relation to *n*. (a) Number of generations, *gen*, to converge (locate at least one maximum). (b) Average affinity of the population after convergence.
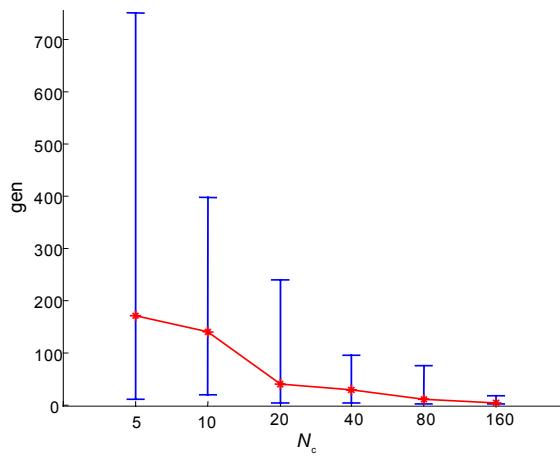
Fig. 13: CLONALG sensitivity to N_c.

(a)   (b)
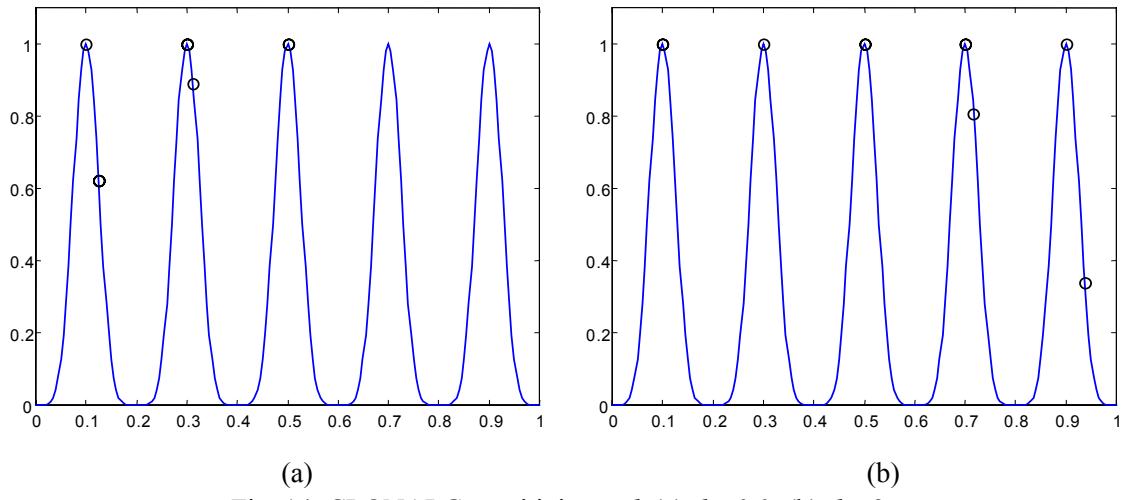
Fig. 14: CLONALG sensitivity to d. (a) $d = 0.0$. (b) $d = 2$.
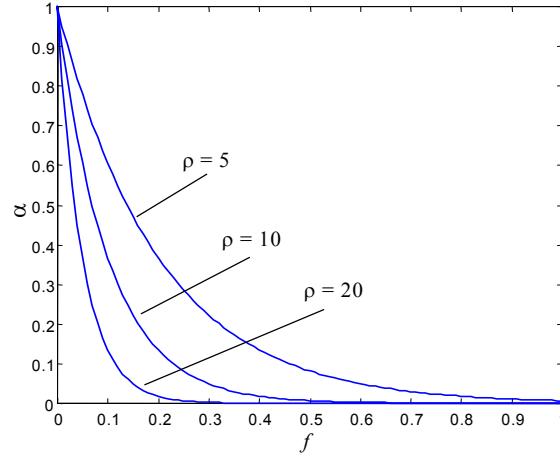
Fig. 15: Relationship between the mutation rate α and the antigenic affinity f.

APPENDIX I

PSEUDOCODE: PATTERN RECOGNITION CASE

See Section V.A for a description.

Input: **Ab,Ag,**gen,n,d,L,β
Output: **Ab**ₘ

```
F := 0;
for t = 1 to gen,
    for j = 1 to M,                                  % Step 1
        f(j,:)    := affinity(Ab,Ag(j,:));           % Step 2
        Abₙ(j,:) := select(Ab,f(j,:),n);            % Step 3
        C(j,:)    := clone(Abₙ,β,f(j,:));           % Step 4
        C*(j,:)   := hypermut(C,f(j,:));             % Step 5
        f*(j,:)   := affinity(C*(j,:),Ag(j,:));      % Step 6
        Ab*       := select(C*,f*(j,:),1);           % Step 7
        Abₘ(j,:) := insert(Abₘ(j,:),Ab*);          % Abₘ ← Ab* if f(Ab*) > f(Abₘ(j,:))
        Ab_d      := generate(d,L);                  % Generate d antibodies of length L
        Ab_r      := replace(Ab_r,Ab_d,f);           % Step 8
    end;
end;
```

APPENDIX II

PSEUDOCODE: OPTIMIZATION CASE

See Section V.B for a description.

Input: **Ab,**gen,n,d,L,β
Output: **Ab,f**

```
for t = 1 to gen,
    f        := decode(Ab);          % Step 2
    Abₙ     := select(Ab,f,n);      % Step 3
    C        := clone(Abₙ,β,f);     % Step 4
    C*       := hypermut(C,f);        % Step 5
    f        := decode(C*);           % Step 6
    Abₙ     := select(C*,f,n);       % Step 7
    Ab       := insert(Ab,Abₙ);      % Ab ← Abₙ
    Ab_d     := generate(d,L);        % Randomly generate d antibodies of length L
    Ab       := replace(Ab,Ab_d,f);   % Passo 8
end;
f := decode(Ab);
```

Note: Function decode is supposed to decode **Ab** and evaluate $g(\cdot)$ for these decoded values.

APPENDIX III

ANALYSIS OF THE ALGORITHM

Analysis of an algorithm refers to the process of deriving estimates for the time and space needed to execute the algorithm. *Complexity of an algorithm* refers to the amount of time and space required to execute it [41], [42]. Determining the performance parameters of a computer program is a difficult task and depends on a number of factors such as the computer being used, the way the data are represented, how and with which programming language the code is implemented. In this section we will present a general evaluation of the complexity of the CLONALG algorithm, taking into account the computational cost per generation and its memory (space) requirements, for the pattern recognition and optimization versions (see Sections V.A and V.B).

We can measure the time required by an algorithm by counting the maximum number of instructions executed, which is proportional to the maximum number of times each loop is executed. Regardless the affinity function, we use parameters that characterize the computations performed, like the available antibody repertoire dimension $N \times L$, the total number of clones N_c, the amount M of antigens to be recognized and the amount n of selected antibodies for reproduction.

The proposed CLONALG algorithm has three main processing steps: 1) determining the affinity of the antibodies (Steps 2 and 6), 2) selecting (and re-selecting) the n highest affinity antibodies (Steps 3 and 7), and 3) hypermutating the population **C** (Step 5).

The usual way of selecting n individuals from a population is by sorting the affinity (fitness) vector f, and then extracting the n first elements of the sorted vector. According to [43], this can be performed in $O(n)$ time. This way, the computational time required in Steps 3 and 7, selection and re-selection phases, is $O(N)$ and $O(N_c)$ in the worst cases, where $n = N$ and $n = N_c$, respectively. Mutating the N_c clones demands a computational time of the order $O(N_c L)$ [29]. By summing up the computational time required for each of these steps, it is possible to determine the total computational time of the algorithm. In the pattern recognition case, these steps have to be performed for each of the M antigens, hence the computational time of the whole process comes preceded by a multiplying factor of M. On the other hand, if the algorithm is to be applied to a multimodal optimization problem with $n = N$, then the selection processes can be suppressed from the algorithm, reducing its computational time by $O(N + N_c)$ in Steps 3 and 7.

In all cases (pattern recognition and optimization), the required memory to run the algorithm is proportional to the dimension N of vector $\boldsymbol{f}$, plus the number N_c of generated clones, plus the dimension of matrices $\mathbf{Ab}_{\{n\}}$ ($\mathbf{Ab}_{\{m\}} \in S^{m \times L}$), $\mathbf{C}$ ($\mathbf{C} \in S^{Nc \times L}$), and $\mathbf{Ab}$ ($\mathbf{Ab} \in S^{N \times L}$).

The computational complexity of the CLONALG algorithm is summarized in Table I.

TABLE I

COMPUTATIONAL COMPLEXITY OF THE ALGORITHM

	Runtime	*Required Memory*
Pattern Recognition	$O(M(N + N_cL))$	
General Optimization	$O(N + N_cL)$	$\propto N + L(m + N_c + N)$
Multimodal Optimization	$O(N_cL)$	