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 (Ag's) are selected to proliferate. The selected cells are subject to an affinity maturation process, which improves their affinity to the selective Ag's. This paper proposes 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 derived primarily 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 in relation to the user-defined parameters is given. CLONALG is also contrasted with evolutionary 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*—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 [1]–[6]. AIS uses ideas gleaned from immunology in order to develop adaptive systems capable of performing a wide range of tasks in various areas of research.

In this paper, 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 useful computational tools. The algorithm to be presented focuses on a systemic view of the immune system and does not take into account cell–cell interactions. It is not our concern to model exactly any biological phenomenon, but to show that some basic immune principles can help us to not only better understand the immune system itself, but also to solve complex engineering tasks.

Initially, the algorithm is proposed and evaluated 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 an antibody (Ab), so that every element of our artificial immune system will be generically called an Ab.

Manuscript received December 5, 2000; revised September 7, 2001. The work of L. N. de Castro was supported by FAPESP under Proc. 98/11333-9 and the work of F. J. Von Zuben was supported in part by FAPESP under Proc. 98/09939-6 and in part by CNPq under Proc. 300910/96-7.

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

Publisher Item Identifier S 1089-778X(02)06065-4.

First, we 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 we show that the same algorithm, with few modifications, is suitable for solving multimodal and combinatorial optimization tasks. This work also contains a discussion relating the proposed clonal selection algorithm, named CLONALG, with well-known evolutionary algorithms (EAs).

The paper is organized as follows. Section II reviews the clonal selection principle and the affinity maturation process from an immunological standpoint. Section III reviews the basics of EAs to perform multimodal search and Section IV characterizes clonal selection as an evolutionary process. Section V briefly discusses a formalism to model immune cells, molecules, and their interactions with antigens (Ag's). Section VI introduces and evaluates the two versions of the proposed algorithm, while Section VII contains the sensitivity analysis of the algorithm in relation to the user-defined parameters. Section VIII discusses the main properties of the algorithm in theoretical and empirical terms. Section IX concludes the paper.

II. Clonal Selection Theory

Any molecule that can be recognized by the adaptive immune system is known as an Ag. When an animal is exposed to an Ag, some subpopulation of its bone-marrow-derived cells (B lymphocytes) responds by producing Ab's. Ab's are molecules attached primarily to the surface of B cells whose aim is to recognize and bind to Ag's. Each B cell secretes a single type of Ab, which is relatively specific for the Ag. By binding to these Ab's and with a second signal from accessory cells, such as the T-helper cell, the Ag stimulates the B cell to proliferate (divide) and mature into terminal (nondividing) Ab 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.

B cells, in addition to proliferating and 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 plasma cells capable of producing high-affinity Ab's, preselected for the specific Ag that had stimulated the primary response. Fig. 1 depicts the clonal selection principle.

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

1) proliferation and differentiation on stimulation of cells with Ag's;
2) generation of new random genetic changes, expressed subsequently as diverse Ab patterns, by a form of accel-

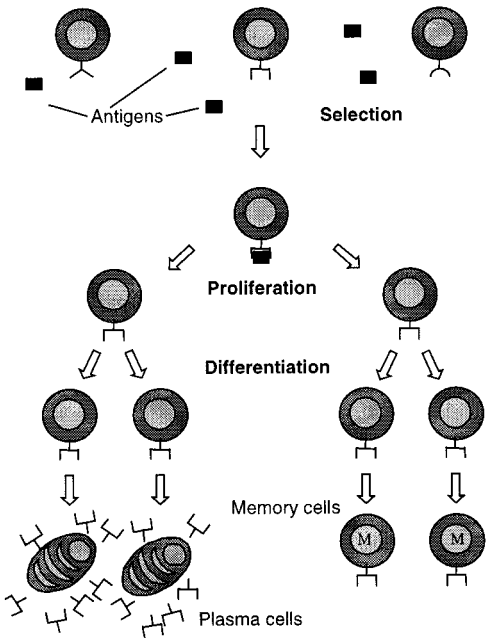

Fig. 1. Clonal selection principle.

erated somatic mutation (a process called affinity maturation);

3) estimation 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 Ag. In our use of the clonal selection theory for the solution of practical problems, we do not intend to maintain a large number of candidate solutions, but to keep a small set of best individuals. A clone will be created temporarily 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 Ag repeatedly during its lifetime. The initial exposure to an Ag that stimulates an adaptive immune response is handled by a small number of low-affinity B cells, each producing an Ab type of different affinity. The effectiveness of the immune response to secondary encounters is enhanced considerably by the presence of memory cells associated with the first infection, capable of producing high-affinity Ab's 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 higher after each infection. This is an intrinsic scheme of a reinforcement learning strategy [9], where the interaction with the environment gives rise to the continuous improvement of the system capability to perform a given task.

To illustrate the adaptive immune learning mechanism, consider that an antigen $\mathbf{Ag}_1$ is introduced at time zero and it finds a few specific Ab's within the animal (see Fig. 2). After a lag phase, the Ab against antigen $\mathbf{Ag}_1$ appears and its concentration rises up to a certain level and then starts to decline (primary response). Consider at this point the exposition to an antigen $\mathbf{Ag}_2$ not correlated with antigen $\mathbf{Ag}_1$. Then, no specific Ab is present and the Ab response will be similar to that obtained in the case of $\mathbf{Ag}_1$ [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 $\mathbf{Ag}_1$ present a faster and more efficient secondary response not only to $\mathbf{Ag}_1$, but also to any structurally related antigen $\mathbf{Ag}_1'$. 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 (computational) intelligence fields, like neural networks [16].

Some characteristics of the associative memories are particularly interesting in the context of AIS:

1) the stored pattern is recovered through the presentation of an incomplete or corrupted version of the pattern;
2) 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 synthesis of Ab's with higher antigenic affinities (see the affinity maturation section). Moreover, a dose of Ag substantially lower than that required to initiate a primary response may cause a secondary response.

Some authors [17], [18] have 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 Ab 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 Ag specific cell has increased. Others argued [14] that whether or not 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 Ag's.

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]:

1) repeated exposure to an Ag;
2) affinity maturation of the receptor molecules (see Section II-B);
3) low-grade chronic infection;
4) cross-reactivity.

B. Affinity Maturation

In a T-cell-dependent immune response, the repertoire of Ag-activated B cells is diversified basically by two mechanisms: 1) hypermutation and 2) receptor editing [20]–[23].

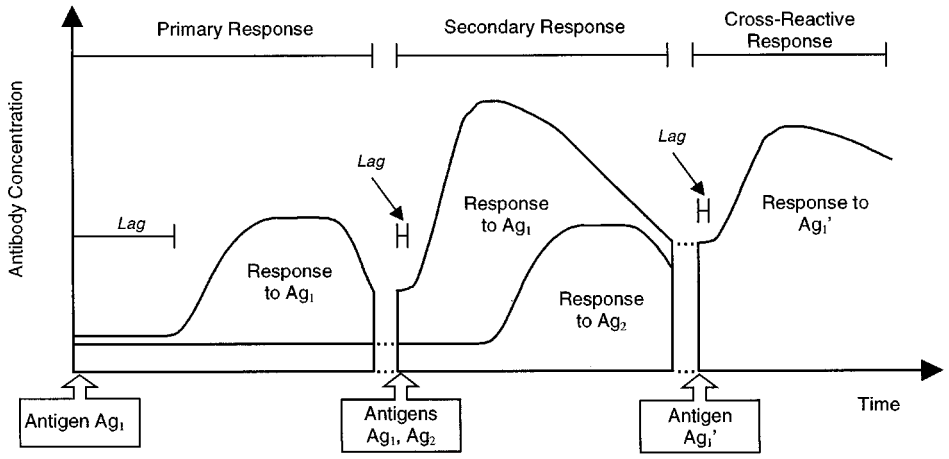

Fig. 2. Primary, secondary, and cross-reactive immune responses. After an Ag has been seen once (primary response), subsequent encounters with the same Ag or a related one (cross reaction) will lead to a faster and stronger response (secondary response).

Ab's 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 Ag-binding sites of the Ab molecules 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 Ab. These higher affinity variants 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 [20]–[22] 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 Ag-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 a particular Ab $(\mathbf{Ab}_1)$ is selected during a primary response, then point mutations allow the immune system to explore local areas around $\mathbf{Ab}_1$ by making small steps toward an Ab with higher affinity, leading to a local optima $(\mathbf{Ab}_1^*)$. Because mutations with lower affinity are lost, the Ab's tend to go up the hill. Receptor editing allows an Ab to take large steps through the landscape, landing in a locale where the affinity might be lower $(\mathbf{Ab}_2)$. However, occasionally the leap will lead to an Ab on the side of a hill where the climbing region is more promising $(\mathbf{Ab}_3)$, reaching the global optimum. From this locale, point mutations can drive the Ab to the top

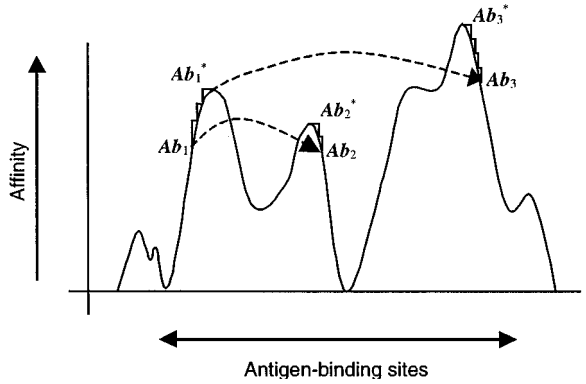

Fig. 3. Schematic representation of shape space for Ag-binding sites. Somatic mutations guide to local optima, while receptor editing introduces diversity, leading to possibly better candidate receptors.

of the hill $(\mathbf{Ab}_3^*)$. 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. This may yield the same result as the process of receptor editing, i.e., a broader search for the global optimum of the Ag-binding site.

1) 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 lead to poorer or nonfunctional Ab's. 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 Ab receptors, however, hypermutation may become inactive [17], [20], generally in a gradual manner.

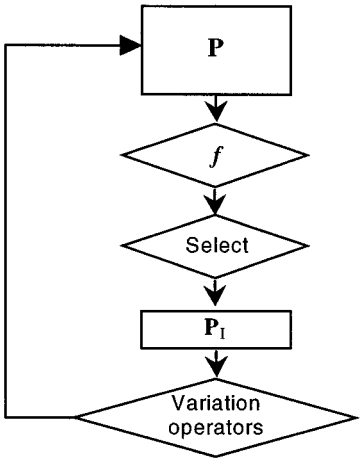

Fig. 4. Computational procedure for a standard EA, where $\mathbf{P}$ represents the population at each generation, f is the vector containing the fitness values for all members of the population, and $\mathbf{P_I}$ is an intermediate population.

III. Evolutionary Algorithms and Niching Methods

As will be seen in Section VIII, CLONALG has many properties in common with the well-known EAs [24]–[26], and can be applied to the same settings. Fig. 4 depicts the block diagram for a standard EA. 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, or the population as a whole is taken as the solution. Selection and variation operators (e.g., crossover, mutation, inversion) guide the population toward optima.

Niching methods extend EAs to domains that require the location and maintenance of multiple solutions, such as multimodal and multiobjective function optimization [26], [27]. 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 or not these solutions have identical or distinct values (fitnesses).

Fitness sharing, first introduced in [28], is a fitness scaling mechanism that alters only the fitness assignment stage of an EA. 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 the most promising regions of the fitness landscape. The number of individuals residing near any peak is proportional to the height of that peak. Sharing works by derating the fitness of each individual in the population by an amount related to the number of similar individuals. Specifically an individual shared fitness f_s is given by

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

where $\mathrm{sh}(\cdot)$ is the sharing function given by (2)

$$\mathrm{sh}(d) = \begin{cases} 1 - \left(\dfrac{d}{\sigma_{\mathrm{share}}}\right)^{\alpha}, & \text{if } d < \sigma_{\mathrm{share}}, \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

where α is a constant that regulates the shape of the sharing function (usually set to one) and σ_{share} represents a threshold of dissimilarity. A sharing evolutionary algorithm $\mathrm{EA_{sh}}$ can distinguish its niches by employing a genotypic or phenotypic distance metric.

IV. Clonal Selection as an Evolutionary Process

The clonal selection functioning of the immune system can be interpreted as a remarkable microcosm of Charles Darwin's theory of evolution, with the three major principles of repertoire diversity, genetic variation, and natural selection [29]. Repertoire diversity is evident in that the immune system produces far more Ab's than will be effectively used to bind to an Ag. In fact, it appears that the majority of Ab's 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 Ab's and selection occurs such that only Ab's able to bind successfully to an Ag will reproduce and be maintained as memory cells.

The similarity between adaptive biological evolution and the production of Ab's is even more striking when one considers that the two central processes involved in the production of Ab's, genetic recombination and mutation, are the same ones responsible for the biological evolution of species. The recombination and editing of immunoglobulin genes underlies the large diversity of the Ab population and the mutation of these genes together with selection serves as a fine-tuning mechanism (see Section II-B). In sexually reproducing species, for example, recombination of gene segments from parent chromosomes, along with mutation, are involved in providing variation and diversity of the species. Selection, then, is responsible for purging those individuals that are less fit to the environment while selecting the fittest ones [24]. Thus, cumulative blind variation and natural selection, which over many millions of years resulted in the emergence of mammalian species, still remain crucial in the day-by-day ceaseless battle to survival of these species through their immune systems. It should, however, be noted that recombination of immunoglobulin genes involved in the production of Ab's 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 [29]. The clonal selection algorithm (CLONALG), to be described further in the paper, aims at demonstrating that this cumulative blind variation, based only upon an affinity proportionate mutation along with a selective pressure, can generate high-quality solutions to complex problems.

V. Shape-Space Model

The shape-space model aims at quantitatively describing the interactions among Ag's and Ab's (Ag-Ab) [30]. The set of fea-

tures that characterize a molecule is called its generalized shape. The Ag-Ab codification (binary or real-valued) determines their spatial representation and a distance measure is used to calculate the degree of interaction between these molecules. Mathematically, the generalized shape of a molecule (m), either an Ab or an Ag, can be represented by a set of L attributes directly associated with coordinate axes such that $m = \langle m_L, \ldots, m_2, m_1 \rangle$ can be regarded as a point in an L-dimensional real-valued shape space $S, m \in S^L \subseteq \Re^L$. The precise physical meaning of each attribute is not relevant to the development of computational tools. In this paper, we consider binary (or integer) strings to represent the molecules. Ag's and Ab's assume the same length L. The length and cell representation depends upon each problem, as will be seen in Section VI.

VI. Computational Aspects of the Clonal Selection Principle

After discussing the clonal selection theory and the affinity maturation process, the development and implementation of CLONALG is straightforward. The main immune aspects taken into account to develop the algorithm are: 1) maintenance of a specific memory set; 2) selection and cloning of the most stimulated Ab's; 3) death of nonstimulated Ab's; 4) affinity maturation; and 5) reselection of the clones proportionally to their antigenic affinity, generation, and maintenance of diversity.

CLONALG is composed basically of two repertoires (populations) of strings: a set of antigens **Ag** and a set of anitbodies **Ab**. The set **Ab** can be decomposed into several subsets according to the application under study (pattern recognition or optimization). Consider the following notation, where boldface expressions (e.g., **Ag** and **Ab**) indicate matrices, boldface italic letters (e.g., $\boldsymbol{Ag}_j$) indicate vectors, S corresponds to a coordinate axis in the shape space, and subindexes within brackets indicate cardinality (e.g., $\mathbf{Ab}_{\{m\}}$).

1) **Ab**: Available Ab repertoire that can be decomposed into several different subsets $(\mathbf{Ab} \in S^{N \chi L}, \mathbf{Ab} = \mathbf{Ab}_{\{r\}} \cup \mathbf{Ab}_{\{m\}}, r + m = N)$.
   a) $\mathbf{Ab}_{\{m\}}$: Memory Ab repertoire $(\mathbf{Ab}_{\{m\}} \in S^{m \chi L}, m \leq N)$.
   b) $\mathbf{Ab}_{\{r\}}$: Remaining Ab repertoire $(\mathbf{Ab}_{\{r\}} \in S^{r \chi L}, r = N - m)$.
   c) $\mathbf{Ab}^j_{\{n\}}$: n: Ab's from **Ab** with the highest affinities to $\boldsymbol{Ag}_j (\mathbf{Ab}^j_{\{n\}} \in S^{n \chi L}, n \leq N)$.
   d) $\mathbf{Ab}_{\{d\}}$: Set of d new Ab's that will replace d low-affinity Ab's from $\mathbf{Ab}_{\{r\}} (\mathbf{Ab}_{\{d\}} \in S^{d \chi L}, d \leq r)$.
2) $\mathbf{Ag}_{\{M\}}$: Population of M Ag's to be recognized $(\mathbf{Ag}_{\{M\}} \in S^{M \chi L})$.
3) $\boldsymbol{f}_j$: Vector containing the affinity of all Ab's in relation to the antigen $\boldsymbol{Ag}_j (\boldsymbol{f}_j \in \Re^N_+)$.
4) $\mathbf{C}^j$: Population of N_c clones generated from $\mathbf{Ab}^j_{\{n\}} (\mathbf{C}^j \in S^{Nc \times L})$. After the maturation (hypermutation) process, the population $\mathbf{C}^j$ is termed $\mathbf{C}^{j*}$.
5) $\boldsymbol{Ab}^*_j$: Candidate from $\mathbf{C}^{j*}$ to enter the pool of memory Ab's.

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

A. Pattern Recognition

In the pattern recognition case, an explicit Ag population $\mathbf{Ag}_{\{M\}}$ with cardinality M is available for recognition. Without loss of generality, it is assumed that $m \geq \boldsymbol{M}$.

The CLONALG algorithm can be described as follows.

1) Randomly choose an antigen $\boldsymbol{Ag}_j$ $(\boldsymbol{Ag}_j \in \mathbf{Ag}_{\{M\}})$ and present it to all Ab's in the repertoire $\mathbf{Ab} = \mathbf{Ab}_{\{r\}} \cup \mathbf{Ab}_{\{m\}} (r + m = N)$.
2) Determine the vector $\boldsymbol{f}_j$ that contains the affinity of $\boldsymbol{Ag}_j$ to all the N Ab's in **Ab**.
3) Select the n highest affinity Ab's from **Ab** to compose a new set $\mathbf{Ab}^j_{\{n\}}$ of high-affinity Ab's in relation to $\boldsymbol{Ag}_j$.
4) The n selected Ab's 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 Ab's.
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 $\boldsymbol{f}^*_j$ of the matured clones $\mathbf{C}^{j*}$ in relation to antigen $\boldsymbol{Ag}_j$.
7) From this set of mature clones $\mathbf{C}^{j*}$, reselect the one with highest affinity $(\boldsymbol{Ab}^*_j)$ in relation to $\boldsymbol{Ag}_j$ to be a candidate to enter the set of memory antibodies $\mathbf{Ab}_{\{m\}}$. If the antigenic affinity of this Ab in relation to $\boldsymbol{Ag}_j$ is larger than its respective memory Ab, then $\boldsymbol{Ab}^*_j$ will replace this memory Ab.
8) Finally, replace the d lowest affinity Ab's from $\mathbf{Ab}_{\{r\}}$, in relation to $\boldsymbol{Ag}_j$, by new individuals in $\mathbf{Ab}_{\{d\}}$.

Fig. 5 depicts the block diagram for the pattern recognition version of CLONALG, including the identification of the eight steps of the algorithm.

After presenting all the M Ag's 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 Ab's were sorted in ascending order after Step 3, so that the number of clones generated for all these n selected antibodies was given by

$$N_c = \sum_{i=1}^{n} \text{round}\left(\frac{\beta \cdot N}{i}\right) \tag{3}$$

where N_c is the total number of clones generated for each of the Ag's, β is a multiplying factor, N is the total number of Ab's, and $\text{round}(\cdot)$ is the operator that rounds its argument toward the closest integer. Each term of this sum corresponds to the clone size of each selected Ab, e.g., for $N = 100$ and $\beta = 1$, the highest affinity Ab $(i = 1)$ will produce 100 clones, while the second highest affinity Ab produces 50 clones, and so on.

In order to evaluate the CLONALG capability to perform learning and memory acquisition, it was applied to a binary character recognition task. The goal is to demonstrate that a cumulative blind variation together with selection can produce individuals with increasing affinities (maturation of the immune

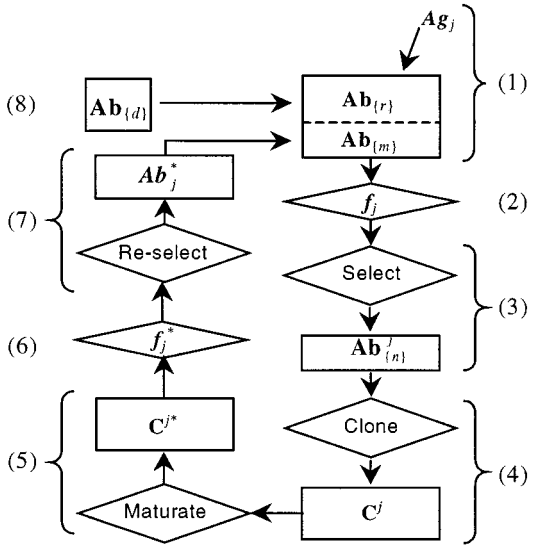

Fig. 5. Computational procedure for CLONALG: pattern recognition version.

response). In this case, it is assumed that the Ag population to be learned is represented by a set of eight binary characters. These characters are the same ones originally proposed by Lippmann [31] in a different context. Each character is represented by a bitstring of length $L = 120$ (the resolution of each picture is 12×10). The Ab 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 $N_{\text{gen}} = 50, n = 5, d = 0.0$, and $\beta = 10$.

The original characters (Ag's) are depicted in Fig. 6(a). Fig. 6(b) illustrates the initial memory set, and Fig. 6(c)–(e) represents the maturation of the memory set (immune response) through generations. The affinity measure takes into account the Hamming distance (D) between an antigen $\boldsymbol{Ag}_j$ and an antibody $\boldsymbol{Ab}_k$, according to

$$D = \sum_{i=1}^{L} \delta_i, \quad \text{where } \delta_i = \begin{cases} 1, & \text{if } \mathrm{Ab}_{ki} \neq \mathrm{Ag}_{ji} \\ 0, & \text{otherwise} \end{cases}. \quad (4)$$

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. Although it is not yet implemented in the current version of CLONALG, a partial matching might allow us to define an affinity threshold ε for recognition. This threshold could be decisive to determine the final size and configuration of the memory set.

B. Optimization

The CLONALG reproduces those individuals with higher affinities, introducing blind variation and selecting their improved maturated progenies. This strategy suggests that the algorithm performs a greedy search, where single members will be optimized locally (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 optimization tasks, particularly multimodal optimization.

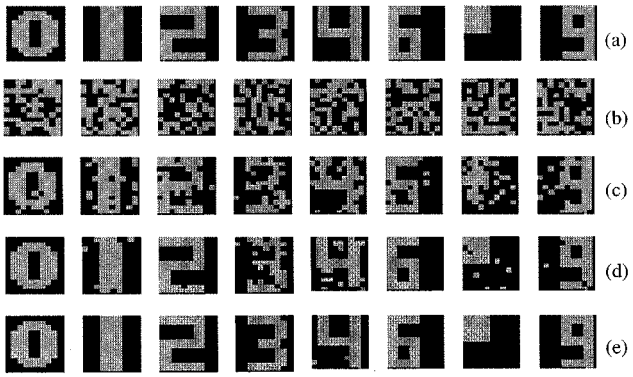

Fig. 6. CLONALG applied to a pattern recognition problem. (a) Patterns to be learned or input patterns (Ag's). (b) Initial memory set. (c) Memory set after 50 cell generations. (d) Memory set after 100 cell generations. (e) Memory set after 200 cell generations.

A few modifications, as described below, have to be made in CLONALG in order to accomplish optimization tasks.

1) In Step 1, there is no explicit Ag population to be recognized, but an objective function $g(\cdot)$ to be optimized (maximized or minimized). This way, an Ab affinity corresponds to the evaluation of the objective function for a given Ab, so that each antibody $\boldsymbol{Ab}_i$ represents an element of the input space. In addition, as there is no specific Ag population to be recognized, the whole Ab population $\mathbf{Ab}$ will compose the memory set and, hence, it is no longer necessary to maintain a separate memory set $\mathbf{Ab}_{\{m\}}$.
2) In Step 7, n Ab's are selected to compose the set $\mathbf{Ab}$, instead of selecting the single best individual $\boldsymbol{Ab}^*_j$.

If the optimization process aims at locating multiple optima within a single population of Ab's, then two parameters may assume default values.

1) Assign $n = N$, i.e., all Ab's from the population $\mathbf{Ab}$ will be selected for cloning in Step 3.
2) The affinity proportionate cloning is not necessarily applicable, meaning that the number of clones generated for each of the N Ab's may be the same, and (3) becomes

$$N_c = \sum_{i=1}^{N} \mathrm{round}(\beta \cdot N). \quad (5)$$

The second statement has the implication that each Ab 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 value of the objective function) will only be accounted to determine the hypermutation rate for each Ab, which is still proportional to their affinity. [Note: In order to maintain the best Ab's for each clone during evolution, it is possible to keep one original (parent) Ab for each clone unmutated during the maturation phase (Step 5)].

Fig. 7 presents the block diagram of the clonal selection algorithm (CLONALG) when adapted to be applied to optimization tasks. The pseudocode for this algorithm is described in Appendix I. Notice that there are slight differences compared to the previous version, depicted in Fig. 5.

Three function optimization tasks were used to evaluate the CLONALG potential to perform multimodal search and a

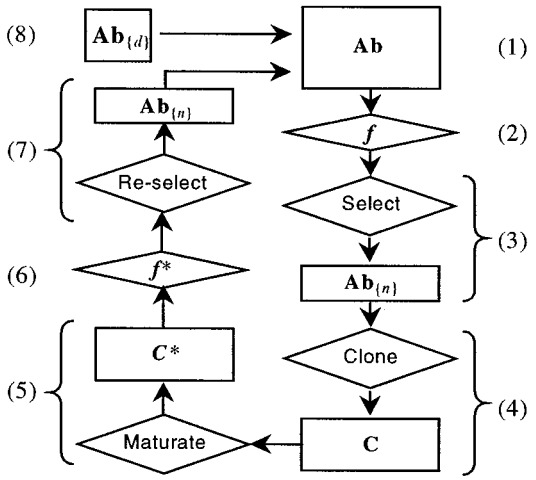

Fig. 7. Computational procedure for CLONALG: optimization version.

30-city instance of the travelling salesman problem (TSP) was used to test its efficacy in solving combinatorial optimization problems.

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

1) $g_1(x) = \sin^6(5\pi x)$;
2) $g_2(x) = 2^{(-2((x-0.1)/0.9)^2}\sin^6(5\pi x)$;
3) $g_3(x,y) = x \cdot \sin(4\pi x) - y \cdot \sin(4\pi y + \pi) + 1$.

The first two unidimensional functions were used in [26], [28], and [32] to evaluate niching methods for genetic algorithms. A discussion relating CLONALG with EAs, including fitness sharing methods, will be presented in Section VIII. In this section, the CLONALG and $\mathrm{EA_{sh}}$ algorithms will be applied to optimize the three functions above and comparisons will be done based upon the quality of the solutions obtained.

Sharing can be implemented using any selection method, but the choice may either increase or decrease the stability of the algorithm [27]. Thus, we chose to implement sharing with a binary tournament selection. In this case, to promote stability at the combination of sharing and tournament selection, we used the continuous updating method, as proposed in [33]. As most applications of genetic algorithms with sharing adopt $\alpha = 1$ and $p_m = 0.0$, we also adopted these values in our simulations. The other parameters employed were $k = 0.6$ as suggested in [34], $p_c = 0.6$ and $\sigma_{\mathrm{share}} = 4$ in all cases. It might be reminded, however, that these parameters were chosen in an *ad hoc* fashion, being only adequate choices, but perhaps not the optimal ones.

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

1) Convert the binary string $m = \langle m_L,\ldots,m_2,m_1\rangle$ from base 2 to base 10: $(\langle m_L,\ldots,m_2,m_1\rangle)_2 = (\sum_{i=0}^{21} m_i \cdot 2^i)_{10} = z'$.
2) Find the corresponding real value for z: $z = z_{\min} + z' \cdot (z_{\max} - z_{\min})/(2^{22} - 1)$, 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.

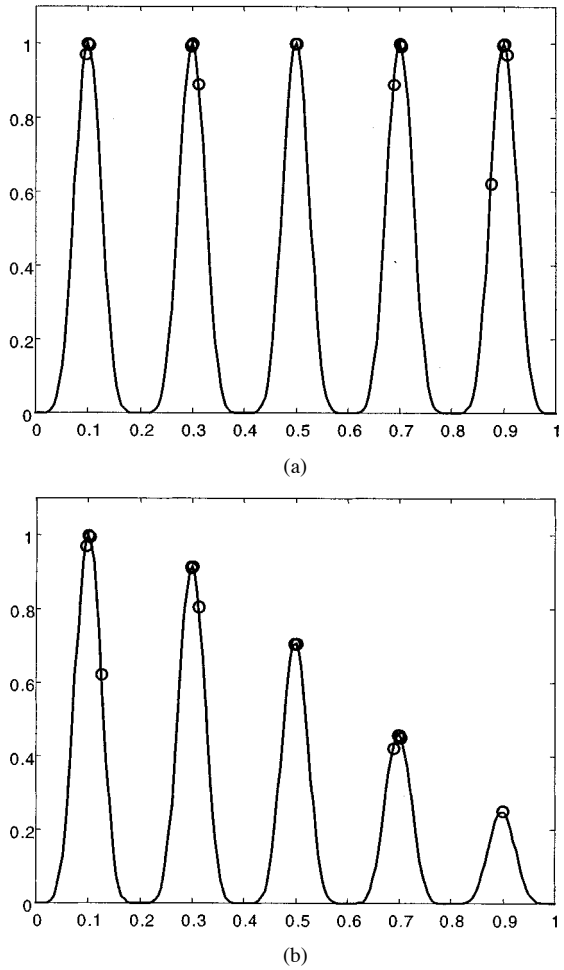

Fig. 8. CLONALG applied to the problem of maximizing functions $g_1(x)$ and $g_2(x)$. (a) $g_1(x) = \sin^6(5\pi x)$. (b) $g_2(x) = 2^{(-2((x-0.1)/0.9)^2}\sin^6(5\pi x)$.

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 a typical result produced by CLONALG after 50 generations. Notice that the solutions (circles) cover all the peaks in both cases. Although there are stochastic steps in the algorithm, the result presented in Fig. 8 does not vary much from one trial to another. The peaks are always determined and the number of nonpeak individuals is always reduced.

The running parameters adopted were:

1) functions $g_1(x)$ and $g_2(x)$: $N_{\mathrm{gen}} = 50, n = N = 50, d = 0.0$, and $\beta = 0.1$ according to (5);
2) function $g_3(x,y)$: $N_{\mathrm{gen}} = 50, n = N = 100, d = 0.0$, and $\beta = 0.1$ according to (5).

Fig. 9 presents the performance of the fitness sharing technique $(\mathrm{EA_{sh}})$ discussed in Section III, when applied to the same problems, and Fig. 10 presents the performance of CLONALG and $\mathrm{EA_{sh}}$ when applied to the problem of maximizing function $g_3(x,y)$. Section VIII will bring general discussions concerning

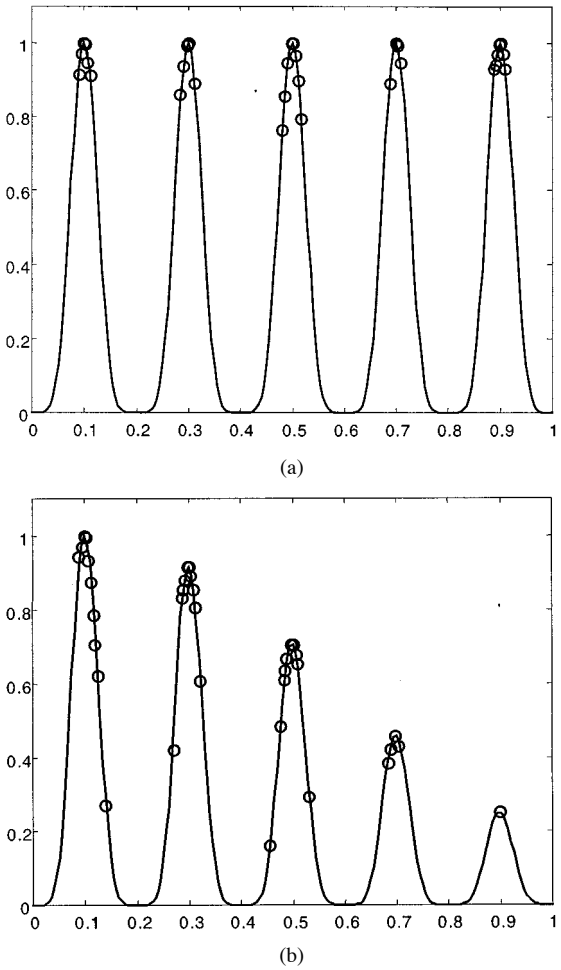

Fig. 9. $\mathrm{EA_{sh}}$ applied to the problem of maximizing functions $g_1(x)$ and $g_2(x), \sigma_{\mathrm{sharg}} = 4$. (a) $g_1(x) = \sin^6(5\pi x)$. (b) $g_2(x) = 2^{(-2((x-0.1)/0.9)^2} \sin^6(5\pi x)$.

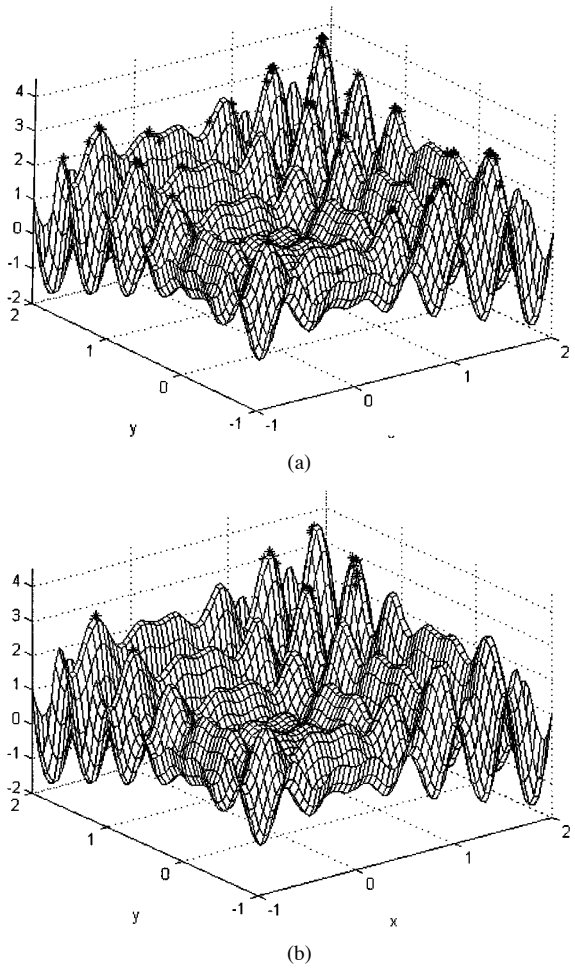

Fig. 10. Maximizing function $g_3(x, y) = x \cdot \sin(4\pi x) - y \cdot \sin(4\pi y + \pi) + 1$. (a) CLONALG. (b) $\mathrm{EA_{sh}}$.

the performance of the algorithms in relation to all the problems tested.

As a last evaluation for the CLONALG algorithm, it was applied to a 30-city TSP. The TSP is a typical case of a combinatorial optimization problem and arises in numerous applications, from very large scale integeration 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 $C = \{1, 2, \ldots, L\}$, represent the possible tours. Each component of the integer vector indexes a city. The inverse of the total length of each tour gives the affinity measure of the corresponding vector. Mutation is performed, in this implementation, by swapping pairs of cities in the Ab's that represent the tours. As proposed in the clonal selection theory, no recombination is performed.

Fig. 11 presents the best solution determined by the CLONALG algorithm, which corresponds to the global optimum [35] after 300 generations. The population size is $N = 300$ Ab's, with $d = 60$ (corresponding to a rate of 20% of newcomers). In this case, low-affinity individuals are allowed to be replaced after each 20 generations. This scheduling is supposed to leave some time for the algorithm to achieve local optima solutions, and then replace the poorer individuals. The other parameters adopted were $N_{\mathrm{gen}} = 300, n = 150$, and $\beta = 2$, according to (3).

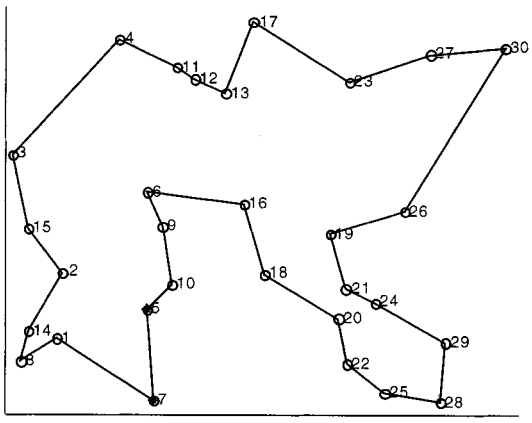

Fig. 11. Best tour determined by the CLONALG algorithm after 300 generations.

VII. 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, which is usual in EAs. However, CLONALG

has three user-defined parameters that influence mainly: 1) the convergence speed; 2) the computational complexity (see Appendix II); and 3) its capability to perform a multimodal search. These parameters are:

1) n (Steps 3 and 7): Number of Ab's to be selected for cloning, giving rise to the population $\mathbf{Ab}_{\{n\}}$ ($\mathbf{Ab}^{j}_{\{n\}}$ in the pattern recognition version);
2) N_c (Step 4): Number of clones generated from $\mathbf{Ab}_{\{n\}}$ [proportional to β according to (3) or (5)];
3) d (Step 8): Amount of low-affinity Ab's to be replaced.

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

A. Sensitivity in Relation to Parameter n

To evaluate the CLONALG sensitivity in relation to n, we fixed $\beta = 1$, resulting in $N_c = 100$ according to (5) and $d = 0.0$. Parameter n takes the values $n = \{2, 4, 6, 8, 10\}$. For $n < N$, if 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 five 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 ten runs of the algorithm for each value of n. It can be inferred 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 (5), and larger values of n imply a higher computational cost to run the algorithm (see Appendix II, Table I).

Fig. 12(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 Ab's that will be located at a maximum of the function.

B. Sensitivity in Relation to Parameter N_c

In order to study the CLONALG sensitivity in 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 five peaks of the function and the average fitness f_{av} of the whole Ab population $\mathbf{Ab}$ was larger than $f_{\text{av}} = 0.92$, where the maximum value is $f = 1.0$. Fig. 13 shows the tradeoff between N_c and the number of generations necessary for convergence (N_{gen}). The results presented are the maximum, minimum, and mean obtained over ten runs of the algorithm for each value of N_c.

It can be seen from the results presented in Fig. 13 that the higher N_c (or β), the faster the convergence, in terms of number of generations. Nevertheless, the computational time per generation increases linearly with N_c, as discussed in Appendix II.

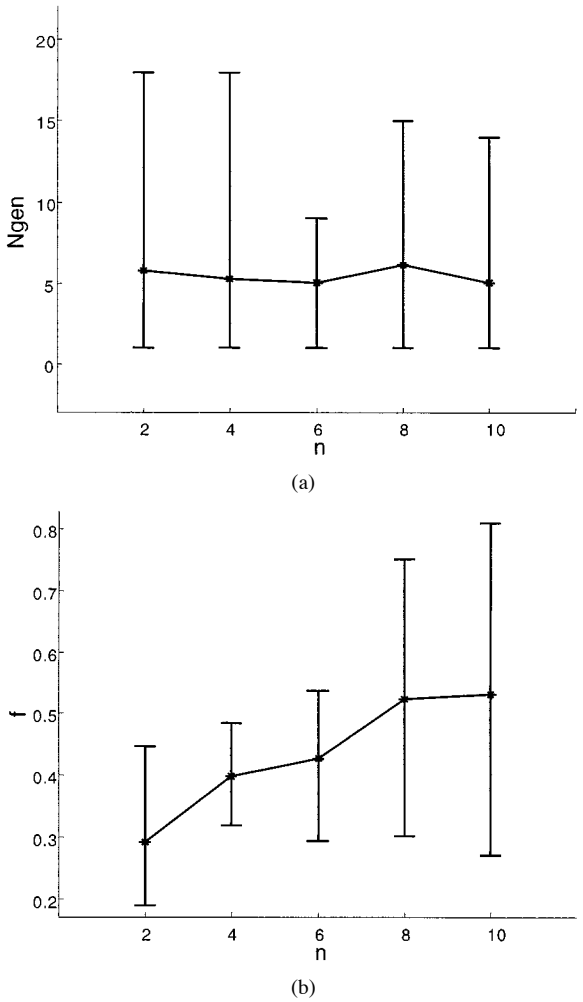

Fig. 12. CLONALG sensitivity in relation to n. (a) Number of generations, Ngen, to converge (locate at least one maximum). (b) Average affinity of the population after convergence.

TABLE I
COMPUTATIONAL COMPLEXITY OF THE ALGORITHM

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

C. Sensitivity in Relation to Parameter d

The number d of low-affinity Ab's to be replaced (Step 8 of the algorithm) is of extreme importance for the introduction and maintenance of diversity in the population and then 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 Ab's, $\mathbf{Ab}_i = \langle 0, \ldots, 0 \rangle, i = 1, \ldots, 10$. The initial affinity of all these Ab's is $f_i = 0, i = 1, \ldots, 10$ 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

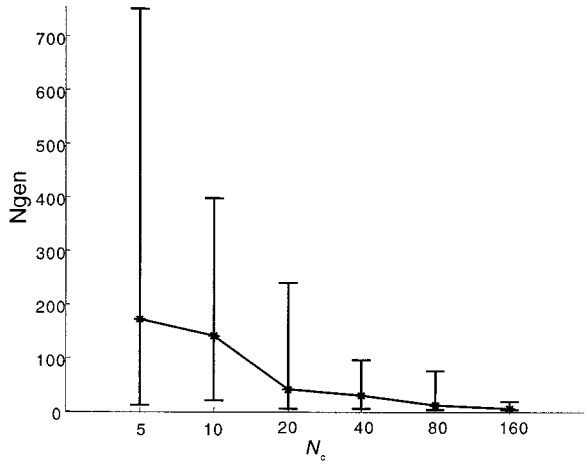

Fig. 13. CLONALG sensitivity in relation to N_c.

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 Ab repertoire). It is also interesting to observe that, even for $d = 0.0$, CLONALG was capable of locating three out of the five peaks of $g_1(x)$.

VIII. Discussion

A. Theoretical Aspects

CLONALG, as proposed in this paper, represents a computational implementation of the clonal selection and affinity maturation principles responsible for describing the behavior of the B cells during an adaptive immune response (see Section II). In both implementations, Figs. 5 and 7, it is assumed the existence of an Ab repertoire $(\mathbf{Ab})$ to be exposed to an antigenic stimulus (which can be caused by an explicit population $\mathbf{Ag}$ to be recognized or a value of an objective function $g(\cdot)$ to be optimized) and those higher affinity Ab's will be selected to generate a population of clones. During proliferation, a few Ab's will suffer somatic mutation proportional to their antigenic affinities and the clones with highest affinity will be selected to compose a memory set. Low-affinity Ab's are replaced, simulating the process of receptor editing.

By comparing CLONALG (Figs. 5 and 7) with the EAs reviewed in Section III (Fig. 4), it is possible to note that the main steps composing the EAs are embodied in CLONALG, allowing us to characterize it as an evolutionary-like algorithm. However, while EAs use a vocabulary borrowed from natural genetics and inspired by Darwinian evolution, the proposed CLONALG makes use of the shape-space formalism, along with immunological terminology to describe Ag-Ab interactions and cellular evolution, as discussed in Section IV. 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 EAs.

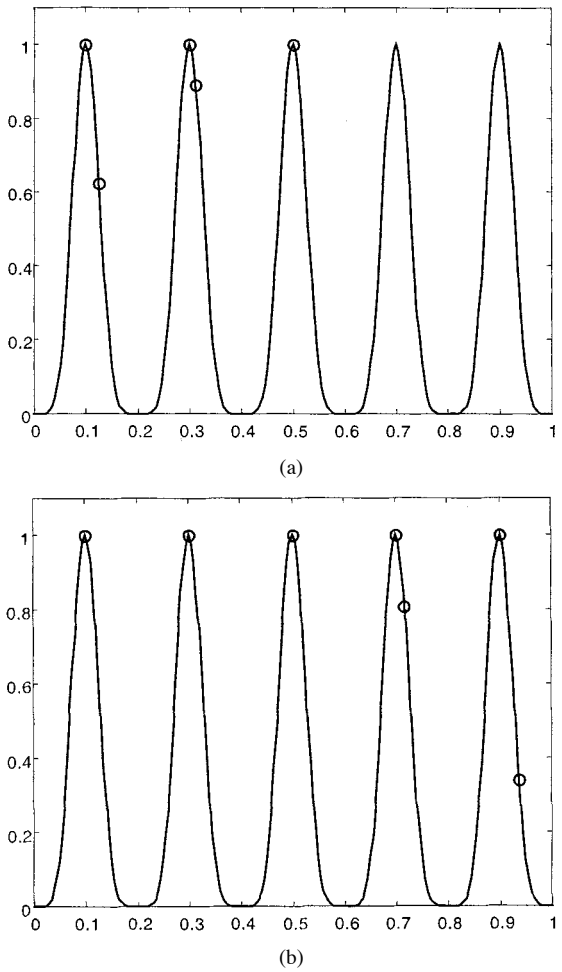

Fig. 14. CLONALG sensitivity in relation to d. (a) $d = 0.0$. (b) $d = 2$.

The proposed clonal selection algorithm can also be characterized as a cooperative and competitive approach, where individual Ab's 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, evolution strategies (ESs) [36], [37] also employ self-adapting variation mechanisms. ESs consider the mutation parameter as the main variation operator. They often use Gaussian mutations and deterministic rules to control the step size [38]. For the CLONALG algorithm, the mutation rate is proportional to the individuals' affinities and may be implemented based on rules like the one described in (6) and illustrated in Fig. 15

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

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

B. Empirical Aspects

The results presented in Section VI 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

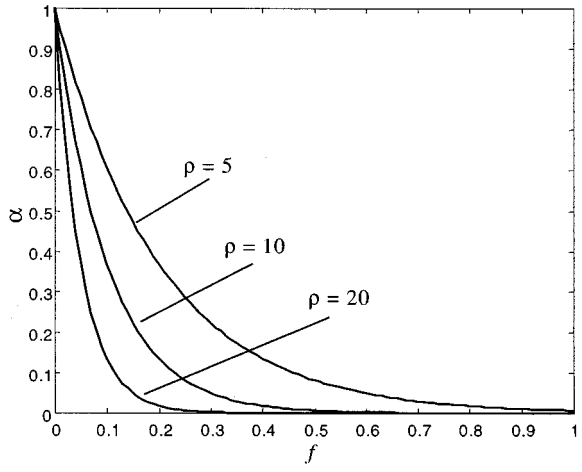

Fig. 15. Tradeoff between the mutation rate α and the antigenic affinity f.

organisms and selfish genes [39]), 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 IV).

By comparing the performance of the proposed CLONALG algorithm for multimodal optimization with the fitness sharing method ($\mathrm{EA_{sh}}$) presented briefly in Section III, we note that the number of nonpeak individuals located by the CLONALG algorithm is much smaller for functions $g_1(x)$ and $g_2(x)$ (see Fig. 8) than those determined by the $\mathrm{EA_{sh}}$ (see Fig. 9). In the case of function $g_3(x, y)$, where the separation among peaks is nonuniform, the fitness sharing overlooked peaks distributed with less uniformity and privileged the highest ones, as illustrated in Fig. 10(b), whilst CLONALG located a higher number of local optima, including the global maximum of the function [see Fig. 10(a)].

While $\mathrm{EA_{sh}}$ has to set parameter σ_{share}, which represents 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 optimization). The sensitivity analysis of the algorithm (see Section VII) 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 related strongly to the convergence speed and computational time required to run the algorithm. As a final aspect of the $\mathrm{EA_{sh}} \leftrightarrow$ CLONALG comparison, the $\mathrm{EA_{sh}}$ 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 II. Each Ab 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 and the receptor editing phase. This way, unlike fitness sharing, the presence of individuals around multiple peaks is not imposed explicitly, being instead an intrinsic aspect of the mechanisms of clonal selection and affinity maturation.

IX. Concluding Remarks

In this paper, we presented a general-purpose algorithm inspired by the clonal selection principle and affinity maturation process, both essential to the adaptive immune response. Two versions of the algorithm, named CLONALG, were derived and subject to a detailed study, including sensitivity analysis and estimation of the computational complexity.

The first version was designed to perform machine-learning and pattern-recognition tasks, where an explicit Ag population, representing a set of input patterns, was available for recognition. Due to its potential to perform parallel search, the algorithm was then adapted to solve optimization problems, emphasizing the multimodal case. In this second version, any Ab was considered a candidate for the optimal solution and the affinity to an Ag was replaced by the corresponding value provided by the function being optimized.

No distinction between the cell and its receptor is made. Thus, the editing and cell replacement processes are equivalent, leading to the introduction of diversity and to a broader exploration of the affinity landscape. Though not considered in the present implementation, several heuristics, e.g., specific types of local search, may be inserted in order to improve the CLONALG performance when applied to particular tasks, such as the TSP exemplified in Section VI.B.

By comparing the proposed algorithm with the standard EA, we note that the CLONALG can reach a diverse set of local optima solutions, while the EA tends to bias the whole population of individuals toward the best candidate solution. Essentially, their encoding schemes and evaluation functions are similar, but their evolutionary search processes differ from the viewpoint of inspiration, vocabulary, and sequence of steps. However, CLONALG still embodies the main steps of any EA.

Empirical comparisons with niching strategies [26], [27], more specifically, with a genetic algorithm with fitness sharing $\mathrm{EA_{sh}}$, demonstrated that CLONALG is capable of locating a larger number of local optima solutions than $\mathrm{EA_{sh}}$. It is also remarkable to note that the CLONALG version for optimization implicitly accounts for the search of multiple solutions. Thus, its computational cost is reduced when compared to an $\mathrm{EA_{sh}}$ that has to explicitly evaluate the degree of similarity among the individuals of the population.

Another important aspect of our model, when compared with the standard EA, is the fact that the CLONALG takes into account the cell affinity, corresponding to the fitness of the individuals, in order to define the proliferation and mutation rates to be applied to each member of the population. ESs [36], [37] and evolutionary programming techniques [40], [41] also employ self-adapting parameters, but based on a mathematical framework that is distinct from the biologically inspired mechanism used by CLONALG.

Appendix I

A. CLONALG—Pseudocode for the Pattern Recognition Version

See Section VI-A for a complete description of the algorithm.

```
Input: Ab, Ag, Ngen, n, d, L, β
Output: Ab_m
for t = 1 to Ngen,
 for j = 1 to M,                              % Step 1
  f(j,:)     := affinity(Ab, Ag(j,:));        % Step 2
  Ab_n(j,:)  := select(Ab, f(j,:), n);        % Step 3
  C(j,:)     := clone(Ab_n, 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
  Ab_m(j,:)  := insert(Ab_m(j,:), Ab*);       % Ab_m ← Ab*
                              if f(Ab*) > f(Ab_m(j,:))
  Ab_d       := generate(d,L);                % Generate d
                              antibodies of length L
  Ab_r       := replace(Ab_r, Ab_d, f);       % Step 8
 end;
end;
```

B. CLONALG—Pseudocode for the Optimization Version

See Section VI-B for a complete description of the algorithm.

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

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

APPENDIX II
COMPLEXITY 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 in the worst case [42], [43]. Determining the performance 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, and how and with which programming language the code is implemented. Here, 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 VI-A and 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 dimension of the available Ab repertoire $N \times L$, the total number of clones N_c, the number M of Ag's to be recognized, and the number n of selected Ab's for reproduction.

The proposed CLONALG algorithm has three main processing steps:

1) determining the affinity of the Ab's (Steps 2 and 6);
2) selecting (and reselecting) the n highest affinity Ab's (Steps 3 and 7);
3) hypermutating the population $\mathbf{C}$ (Step 5).

The usual way of selecting n individuals from a population is by sorting the affinity (fitness) vector $\boldsymbol{f}$ and then extracting the n first elements of the sorted vector. According to [44], this can be performed in $O(n)$ time. This way, the computational time required in Steps 3 and 7, selection and reselection 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 $O(N_cL)$ [26]. 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 Ag's; 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 problems with $n = N$, then the selection processes can be suppressed from the algorithm, reducing its computational time to $O(N_cL)$.

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 $\mathbf{Ab}_{\{m\}}$ ($\mathbf{Ab}_{\{m\}} \in S^{m\times L}$), $\mathbf{C}$ ($\mathbf{C} \in S^{N_c\times L}$), and $\mathbf{Ab}$ ($\mathbf{Ab} \in S^{N\times L}$).

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

REFERENCES

[1] L. N. de Castro and J. Timmis, *Artificial Immune Systems: A New Computational Intelligence Approach*. London, U.K.: Springer-Verlag, 1996.

[2] J. E. Hunt and D. E. Cooke, "Learning using an artificial immune system," *J. Network Comput. Applicat.*, vol. 19, no. 2, pp. 189–212, Apr. 1996.

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

[4] S. A. Hofmeyr and S. Forrest, "Immunity by design: An artificial immune system," in *Proc. Genetic and Evolutionary Computation Conf.*, July 1999, pp. 1289–1296.

[5] L. N. de Castro and F. J. Von Zuben. (1999) Artificial Immune Systems: Part I—Basic Theory and Applications. FEEC/Univ. Campinas, Campinas, Brazil. [Online]. Available: http://www.dca.fee.unicamp.br/~lnunes/immune.html

[6] ——, (2000) Artificial Immune Systems: Part II—A Survey of Applications. FEEC/Univ. Campinas, Campinas, Brazil. [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. New York: Marcel Dekker, 1978, pp. 63–85.

[8] ——, *The Clonal Selection Theory of Acquired Immunity*. Cambridge, U.K.: Cambridge Univ. Press, 1959.

[9] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.
[10] C. A. Janeway, P. Travers, M. Walport, and J. D. Capra, *Immunobiology: The Immune System in Health and Disease*, 4th ed. New York: Garland, 1999.
[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, no. 2, pp. 141–150, Nov. 1997.
[12] P. D. Hodgkin, "Role of cross-reactivity in the development of antibody responses," *Immunologist*, vol. 6, no. 6, pp. 223–226, 1998.
[13] G. L. Ada and G. Nossal, "The clonal selection theory," *Sci. Amer.*, vol. 257, no. 2, pp. 50–57, 1987.
[14] J. Sprent, "T and B memory cells," *Cell*, vol. 76, no. 2, pp. 315–322, Jan. 1994.
[15] D. Mason, "Antigen cross-reactivity: Essential in the function of TCRs," *Immunologist*, vol. 6, no. 6, pp. 220–222, 1998.
[16] S. Haykin, *Neural Networks—A Comprehensive Foundation*, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 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," *Immun. Rev.*, no. 96, pp. 5–22, Apr. 1987.
[18] A. Coutinho, "Beyond clonal selection and network," *Immun. Rev.*, no. 110, pp. 63–87, Aug. 1989.
[19] R. Ahmed and J. Sprent, "Immunological memory," *Immunologist*, vol. 7, no. 1–2, pp. 23–26, 1999.
[20] C. Berek and M. Ziegner, "The maturation of the immune response," *Immun. Today*, vol. 14, no. 8, pp. 400–402, 1993.
[21] A. J. T. George and D. Gray, "Receptor editing during affinity maturation," *Immun. Today*, vol. 20, no. 4, p. 196, 1999.
[22] M. C. Nussenzweig, "Immune receptor editing: Revise and select," *Cell*, vol. 95, no. 7, pp. 875–878, Dec. 1998.
[23] S. Tonegawa, "Somatic generation of antibody diversity," *Nature*, vol. 302, no. 14, pp. 575–581, Apr. 1983.
[24] J. H. Holland, *Adaptation in Natural and Artificial Systems*, 5th ed. Cambridge, MA: MIT Press, 1998.
[25] T. Bäck, D. B. Fogel, and Z. Michalewicz, *Evolutionary Computation 1: Basic Algorithms and Operators*. Bristol, U.K.: Inst. of Physiscs, 2000.
[26] ——, *Evolutionary Computation 2: Advanced Algorithms and Operators*. Bristol, U.K.: Inst. of Physiscs, 2000.
[27] S. W. Mahfoud, "Niching methods for genetic algorithms," Illinois Genetic Algorithms Lab., Univ. Illinois, Urbana, IL, IlliGAL Rep. 95001, May 1995.
[28] D. E. Goldberg and J. Richardson, "Genetic algorithms with sharing for multimodal function optimization," in *Proc. 2nd Int. Conf. Genetic Algorithms*, 1987, pp. 41–49.
[29] G. Cziko, "The immune system: Selection by the enemy," in *Without Miracles*, G. Cziko, Ed. Cambridge, MA: MIT Press, 1995, pp. 39–48.
[30] 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, no. 4, pp. 645–670, Dec. 1979.
[31] R. P. Lippmann, "An introduction to computing with neural nets," *IEEE Acoust., Speech, Signal Processing Mag.*, vol. 2, pp. 4–22, Aro. 1987.
[32] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addisson-Wesley, 1989.
[33] C. K. Oei, D. E. Goldberg, and S.-J. Chang, "Tournament Selection, Niching and the Preservation of Diversity," Illinois Genetic Algorithms Lab., Univ. Illinois, Urbana, IL, IlliGAL Rep. 91011, Dec. 1991.
[34] M. Mitchell, *An Introduction to Genetic Algorithms*. Cambridge, MA: MIT Press, 1998.
[35] P. Moscato and J. F. Fontanari, "Stochastic versus deterministic update in simulated annealing," *Phys. Lett. A*, vol. 146, no. 4, pp. 204–208, 1990.
[36] I. Rechenberg, *Evolutionsstrategie: Optimierung Technischer Systeme Nach Prinzipien der Biologischen Evolution*, Stuttgart: Frommann-Holzboog, 1993.
[37] ——, "Evolution strategy," in *Computational Intelligence Imitating Life*, J. M. Zurada, R. J. Marks II, and C. J. Robinson, Eds. Piscataway, NJ: IEEE Press, 1994, pp. 147–159.
[38] T. Bäck, U. Hammel, and H.-P. Schwefel, "Evolutionary computation: Comments on the history and current state," *IEEE Trans. Evol. Comput.*, vol. 1, pp. 3–17, Apr. 1997.
[39] R. Dawkins, *The Selfish Gene*. London, U.K.: Oxford Univ. Press, 1989.
[40] D. B. Fogel, L. J. Fogel, and J. W. Atmar, "Meta-evolutionary programming," in *Proc. 25th Conf. Signals, Systems, and Computers*, R. Chen, Ed.. San Jose, CA, 1991, pp. 540–545.
[41] D. B. Fogel, *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. Piscataway, NJ: IEEE Press, 1995.
[42] R. Johnsonbaugh, *Discrete Mathematics*, 4th ed. Englewood Cliffs, NJ: Prentice-Hall, 1997.
[43] E. Horowitz and S. Sahni, *Fundamentals of Computer Algorithms*. New York: Pitman, 1978.
[44] M. Fischetti and S. Martello, "A hybrid algorithm for finding the kth smallest of n elements in $O(n)$ time," *Annu. Oper. Res.*, vol. 13, pp. 401–419, 1988.

Leandro N. de Castro (S'96–M'01) received the B.Sc. degree in electrical engineering from the Federal University of Goiás, Goiânia, Brazil, in1996 and the M.Sc. degree in control engineering and the Ph.D. degree in computer engineering from the State University of Campinas (UNICAMP), Campinas, São Paulo, Brazil, in 1998 and 2001, respectively.

He was a Research Associate with the Computing Laboratory at the University of Kent, Canterbury, U.K., from 2001 to 2002. He is currently a Visiting Lecturer witht he School of Computer and Electrical Engineering, UNICAMP. His current research interests include artificial immune systems, artificial neural networks, and evolutionary algorithms.

Dr. de Castro is a Member of the International Neural Network Society and the Brazilian Society on Automation.

Fernando J. Von Zuben (S'92–M'96) received the B.Sc. degree in electrical engineering and the M.Sc. and Ph.D. degrees in automation from the State University of Campinas, Campinas, São Paulo, Brazil, in 1991, 1993, and 1996, respectively.

Since 1997, he has been an Assistant Professor with the Department of Computer Engineering and Industrial Automation, the State University of Campinas. His current research interests include artificial neural networks, artificial immune systems, evolutionary algorithms, nonlinear control systems, and multivariate data analysis.

Dr. Von Zuben is a Member of the International Neural Network Society.