FUNCTION OPTIMIZATION BY THE IMMUNE METAPHOR

SŁAWOMIR T. WIERZCHOŃ

Institute o Computer Science, Polish Academy of Sciences, Ordona 21, 01-267 Warsaw, Poland
stw@ipipan.waw.pl
Department of Computer Science, Technical University of Bialystok, Wiejska 45a, 15-333 Bialystok, Poland

(Received 26 April 2002)

Abstract: The main goal of the immune system is to protect an organism against pathogens. To be able to recognize unknown (*i.e.* never seen) pathogens, the immune system applies a number of methods allowing to maintain sufficient diversity of its receptors. The most important methods are clonal selection and suppression of ineffective receptors. In effect the immune system admits maturation affinity property: during its functioning it continuously improves its ability to recognize new types of pathogens. This idea had found many interesting computer-oriented applications. In this paper a simple and easy to implement algorithm for multi-modal as well as non-stationary functions optimization is proposed. It is based on clonal selection and cells suppression mechanisms. Empirical results confirming its usability for uni-, multi-modal and non-stationary functions optimization are presented, and a review of other immunity-based approaches is given.

Keywords: artificial immune systems, multi-modal optimization, non-stationary functions optimization, immune memory, clonal selection, affinity maturation

1. Introduction

Stochastic iterative search techniques proved to be a useful tool in solving hard optimization problems. Their popularity follows from at least three features, ([1], pp. 1–8): they are easy to implement, they are very general (in a sense that there are no special requirements on the kinds of optimization problem), and lastly, they proved to be satisfactory in finding good, fast solutions to a wide spectrum of hard problems. These techniques can be divided into two main categories: local search and population-based search or evolutionary algorithms (EAs for brevity). While techniques from the first category maintain a single "current best solution", population-based techniques maintain a population of tentative solutions that are modified in a way resembling natural evolution. Genetic algorithms (GAs for short) and evolutionary strategies (ESs), are well-known examples of EAs. In the case of GAs subsequent populations are formed by applying three operators resulting in: (1) repetition of best candidates (through *selection*), (2) exchanging information among the members of the population (through *crossover*) and experimentation (*mutation*), while ESs use only selection and mutation.

An attractive feature of a GA is its exploitative aspect: According to Holland's schema theorem (consult *e.g.* [2] for details) the algorithm assigns an exponentially increasing number of trials to the observed best parts of the search space, which results in a convergence to a single point, being a searched optimum. However, this convergence is not always advantageous, and – as stated by Gaspar and Collard in [3] – it contradicts the basic principle of natural evolution, where a great diversity of different species is observed. Classical GA cannot maintain sufficient population diversity, and it behaves poorly when solving *e.g.* multi-modal optimization problems or when tracing for the optimum of periodic non-stationary functions .

A corollary to the schema theorem is the building block hypothesis (BBH), which states that if two good solutions are properly combined, then an even better solution (with rather high probability) is obtained. Careful analysis of a GA behavior shows that for real-life problems of nontrivial size and difficulty, the contribution of crossover search, responsible for the combination of solutions, is marginal. Park and Carter [4] studied the performance of genetic search and its components in the context of the problem of finding the size of a maximum clique in a graph. They stated that: [...] *for "easy" problems where BBH holds true, finding a near-optimal solution is inherently easy task, and other more efficient algorithms are preferable to the compute-intensive GA. For more "difficult" problems where BBH is not readily applicable, crossover yields only a negligibly small probability of success, while turning into a burden by restricting the number of states that can be visited in the search space within reasonable resource bounds.* This observation was confirmed by Spears [5], who analyzed crossover and mutation in a different context. Treating both operators as disruptive operators, he showed that mutation can provide any level of disruption that crossover can provide. Further, crossover has no advantage over mutation in terms of the amount of exploration that can be performed. In fact, mutation can provide higher levels of disruption and exploration, and this phenomenon explains in a sense successful behavior of ESs.

Recently, a new biologically inspired technique, so-called *artificial immune systems* (AISs), has been proposed to deal with multi-modal functions optimization or non-stationary functions optimization. The learning/adaptive mechanisms used by the natural immune system allow continuous generation of new species of so-called antibodies responsible for detection and destruction of foreign molecules, called antigens or pathogens. Current estimates show that although the human genome contains about 10^5 genes, it is able to produce antibodies repertoire that can recognize at least 10^{16} pathogens. The enormous diversity of the antibodies developed by the immune system is the key to its pathogen recognition capabilities.

Using this idea we propose a novel, simple and efficient, algorithm for solving multi-modal optimization problems. Contrary to popular binary AIS, where pathogens and antibodies are represented as binary strings, this algorithm uses more natural, real-valued representation that speeds up its convergence.

The paper is organized as follows. Section 2 provides a brief overview of the main mechanisms used by the immune system. Current approaches to the multi-modal function optimization are reviewed in Section 3. In Section 4 new algorithm is presented and numerical results are given. Section 5 concludes the paper.

2. Basic facts from immunology

The aim of this section is a brief presentation of the main mechanisms used by the immune system. A reader interested in a deeper review of natural immune system from a mathematical perspective is referred to [6], while computer-oriented treatment of the problem can be found in [7], Part III of [1], or in [8].

The basic building blocks of the immune system are white blood cells, or lymphocytes. There are two major classes of lymphocytes: B-cells, produced in the *bone marrow* in the course of so-called clonal selection (described later), and T-cells, processed in the *thymus*. B-lymphocytes are related to humoral immunity: they secrete antibodies. Among the B-cells are "memory" cells. Living relatively long and "remembering" foreign proteins they constantly re-stimulate the immune response of the organism. On the other hand, T-cells are concerned with cellular immunity: they function by interacting with other cells. T-cells divide into CD4 lymphocytes or helper T-cells, and CD8 lymphocytes, called cytoxic or killer T-cells, that eliminate intracellular pathogens. Generally, helper T-cells activate B-cells promoting their growth and differentiation into an antibody-secreting state. Activated B-cells cut protein antigens into smaller parts (peptides) and present them to killer T-cells. These last cells are responsible for killing virally infected cells and cells that appear abnormal. B- and T-cells see different features of pathogens, and behave quite differently. Thus simulating B- or T-cells we obtain computer systems solving different tasks. Computer systems simulating T-cells are used to broadly understood anomaly detection [9], *e.g.* computer viruses detection [10], while the systems simulating B-cells behavior are oriented towards pattern recognition problems (some of them are reviewed in Section 3.1). In the sequel we will focus on B-cells only.

A lymphocyte has about 10^5 receptors, which are of the same structure. In the case of B-cells, the receptor is an antibody (immunoglobulin) molecule embedded in the membrane of the cell. The specialized portion of the antibody used for identifying other molecules is called *paratope*, while the regions on any molecule that the paratopes can attach to are called *epitopes*. Binding between a paratope and an epitope is based on the complementarity of their shapes. The strength of the bond between an epitope and a paratope is termed the affinity. A lymphocyte is said to be activated if the number of pathogens binding to its receptors exceeds the affinity threshold (and if it obtains special signal from T-cells). To treat the problem of recognizing pathogens by the receptors analytically, Perelson [11] introduced the notion of the shape space. Namely, if n is the set of features influencing the interaction between the molecules – *i.e.* spatial parameters (the length, height, width), charge distribution, *etc.* – then a point in n-dimensional space S is the generalized shape of a molecule. Typically S is a subset of n-dimensional Hamming space, or n-dimensional Euclidean space.

The receptors are constructed from inherited gene segments (libraries) and they come into being in the process of random recombination of segments from different libraries. The process relies upon random selection of a genetic component from each of the libraries. There are many possible combinations of the available components, so the immune system can generate a large number of antibodies even though the libraries contain a limited amount of genetic information. Additionally the libraries evolve in time. This mechanism is responsible for the combinatorial diversity of the receptors.

Clonal selection is another mechanism guaranteeing large diversity of the receptors. When a cell is activated by binding to pathogens, it secretes a soluble form of its receptors and, simultaneously, it clones itself. Clones are not perfect, but they are subjected to somatic mutation (characterized by high mutation rate) that result with children having slightly different receptors than the parent. These new B-cells can also bind to pathogens and if they have a high affinity to the pathogens they in turn will be activated and cloned. The rate of cloning a cell is proportional to its "fitness" to the problem: the fittest cells replicate the most. The somatic mutation guarantees sufficient variation of the set of clones, while selection is provided by competition for pathogens. This mechanism was employed by Hunt and Cooke, [12], to create a learning system, and by Bersini and Varela [13], in solving optimization problems.

Further diversity of the B-cells repertoire is maintained by the replacement of poorly behaving cells. They are simply killed and new, randomly generated, cells are introduced. Approximately five percent of the B-cells is replaced every day by new lymphocytes generated in the bone marrow.

Clonal selection (operating on individuals), stochastic gene selection (operating on genes that determine the specificity of antibodies) and programmed death of cells (called apoptosis) are the main mechanisms providing an exponential number of combinations. Potentially the immune system can produce 10^{15} different receptors, although an estimated number of receptors present in a body at any given time varies between 10^8–10^{12}.

Jerne's hypothesis [14] states that the recognition events between antibodies play a crucial regulatory role in the immune system. Although not verified empirically, this offers a conceptual tool nicely explaining the phenomena of immune memory (its true nature is still unknown to the immunologists), and it is an inspiration for constructing successful computer immune systems (consult [7], [15], or [16]).

The main goal of the immune system is to protect an organism against pathogens. In the AISs a problem with unknown solution is treated as a pathogen, while the solution to this problem is an antibody and the problem relies upon evolving the set of antibodies in such a way they are able to recognize the pathogen or the set of pathogens if there are multiple solutions to the original problem. Recently edited volumes [7] and Part III of [1] review ideas and applications of AISs.

3. Current works on immune-based optimization

Optimization is a subject of an intensive research of the "evolutionary community". This section briefly reviews different approaches to this problem. Since much of the immune approaches use GA as a tool for evolving antibodies we start from a brief overview of GA-based approaches, and next we review papers devoted to the application of immune metaphor in broadly understood optimization. Subsection 3.1

presents only some important approaches to the "genetic" optimization of multi-modal functions while Subsection 3.2 is a state-of-art in the immune-based optimization.

3.1. GA-based approaches

A "classical" technique allowing to explore new areas of the search space without destroying good candidates already acquired is that of sharing and restricted mating proposed in 1987 by Goldberg and Richardson (consult [2] for details). An optimum, or simply a peak in a search space, is treated here as a resource that a sub-population, or species can exploit. The individuals near the peak have to share the resource of that peak. Perceived fitness, *i.e.* the EA's perception of the objective function, depends on the number of individuals located near the peak: the more the individuals the lower the fitness is, as the resource becomes overused, which reduces selective pressure in that area of search space. Apart from the perceived fitness the similarity metric is used to restrict crossover to most similar candidates only. This idea makes two assumptions, however: (a) the number of peaks is known in advance, and (b) the peaks are uniformly distributed throughout the search space. In comparison with "traditional" evolutionary algorithm the time complexity of this algorithm is increased by $O(n^2)$, where n is the population size. This additional time is needed to perform the fitness sharing calculations. Another serious problem with real implementation is that of niche radius: if both assumptions are fulfilled, a simple recipe can be used to estimate the radius. However, there are functions where these assumptions are not satisfied.

Beasley *et al.* [17], introduced a sequential niche technique. It can be treated as a simple extension to existing unimodal techniques as it involves multiple runs of an EA (or other search technique), each finding one maximum. Again, a vital assumption is that the number of interesting optima is known in advance, and that these optima are spread approximately uniformly throughout the search space. A number of problems with this algorithm was reported in [17]: inaccuracy, incompleteness, extra runs, and – again – the problem with niche radius.

To avoid the problems with the assumptions (a) and (b), and to make the implementation less expensive, Spears [18] replaced the distance metric by labels attached to each individual. Each label consists of t tag bits, which allows to distinguish 2^t sub-populations. To attain fitness sharing, the perceived fitness is obtained by dividing the row fitness of each individual by the size of its sub-population. Such an idea appears to be useful in finding the high peaks in the search space. To find lower peaks, additional mechanism restricting mating further had to be used.

3.2. Immune-based approaches

Perhaps the first work investigating potential application of the immune metaphor in solving numerical optimization problems was the paper by Bersini and Varela [13], who proposed immune recruitment mechanism (IRM). It refers to so-called meta-dynamics of the immune system and relying upon continuous generation of new species, *i.e.* new points in the search subspace delimited by the n best points. This algorithm was used for finding global optimum of a function over a fixed domain.

In 1993 Forrest *et al.* [19], and Smith *et al.* [20], showed that an immune system model combined with a genetic algorithm can be used to evolve a set of antibodies that recognize a range of diverse pathogens. In both works the authors focused on a binary immune system where pathogens and antigens were represented as binary string of fixed length. Constructing their algorithm, called the diversity algorithm, or DA, the authors referred to the following properties of the immune systems:

a) pathogens are typically encountered sequentially,
b) an immune system responds with only a subset of its lymphocytes – those that come in contact with pathogen (to mimic this a sample of size σ was chosen randomly without replacement each time a new pathogen has been entered),
c) there is competition for antigens so that the cells that bind with the highest affinity (measured here by the Hamming distance) grow the fastest, and
d) antibodies are improved by somatic mutation.

Careful analysis of the algorithm behavior has shown its two interesting features. First, the DA was able to maintain diverse sub-populations within a given population of individuals. The existence of such sub-populations was insensible on the similarity (measured by the Hamming distance) among pathogens. For small sample size, σ, generalist antibodies were evolved that match all pathogens through the identification of common schemata, and for high σ specialist antibodies could be produced, each of which matches a different pathogen. Second important property was that the process of such sub-populations formation was similar to the fitness sharing. However, the fitness sharing in this new algorithm is implicit in the sense that the algorithm determines the number of peaks dynamically without regard on the number of peak and the distance between peaks.

Hajela and Lee [21], used this last property to design a number of algorithms devoted to solving structural optimization problems and multi-criteria decision problems. To handle constraints, for instance, a standard GA with the DA, playing a role of a *repair algorithm*, was used. The algorithm works as follows. An initial population of individuals (mapped into binary strings of fixed length) is randomly generated, and each individual is evaluated to compute its objective function and a global measure of constraints violation. Next, the subpopulations of feasible and infeasible individuals are identified and each subpopulation is sorted decreasingly with respect to the objective function value. A number of best feasible individuals forms the pathogen population. The infeasible solutions, treated as the antibodies, are modified by the DA to the generalist antibodies with reduced constraint violation level. Finally, the designs from the subpopulation of feasible designs are mixed with multiple copies of the best generalist antibodies to form a population subjected to standard genetic operations of selection, crossover and mutation. Now, the offsprings are evaluated to compute their objective function and a global measure of constraints violation and the whole process repeats till the convergence.

Hightower [22], extended further the DA proposing stochastic gene expression algorithm (SGEA). In his approach a population consists not of entire antibodies but of gene libraries mentioned earlier, and an antibody molecule is constructed by choosing randomly one segment of each library. Because of such construction the phenotype of an individual (*i.e.* expressed antibody strings) does not completely represent its genotype (the total collection of gene segments in the library). Hence, best parts of the search space discovered in one cycle are rather different from best parts identified in the next cycle as random segment selection allows the segments to be temporarily hidden from selection stage of the genetic algorithm. An interesting feature of the SGEA is that the antibodies evolve to maximize the average Hamming distance to the other antibodies in the library. (A careful mathematical analysis of this phenomenon can be found in Oprea [23]). SGEA was used in function optimization problem, the Holland's Royal Road (or HRR) function, and its superior performance over an EA using complete gene expression has been observed. Two other experiments using variants of HRR functions confirmed the former observation. Again, the algorithm admits an implicit fitness sharing allocating antibody genes to multiple pathogens. Hart and Ross [24] used SGEA to produce robust schedules for a dynamic job-shop scheduling problem in which jobs arrive continually, and the environment is subject to change due to practical reasons. In this approach pathogen represents a set of changes that can occur forcing changes in a schedule while antibody represents a schedule.

The problem of multi-modal function optimization was addressed explicitly by Fukuda *et al.* in [25]. Their algorithm refers to the basic mechanism discussed earlier and consists of six steps. The first step is the recognition of antigen, *i.e.* the identification of an optimization problem. Next, memory cells are used to produce antibodies (represented by binary strings of fixed length) that were effective in the past. The affinity of the antibodies to the newly entered pathogens is computed. "Good" solutions are saved as memory cells for the next iterations while the remaining solution candidates are eliminated; this way locally optimal solutions are stored. Next, good antibodies proliferate proportionally to their affinity value, while proliferation of an antibody with extremely high concentration is suppressed, which enables to maintain the diversity of searching directions. Finally, in the place of suppressed antibodies new lymphocytes are produced by using genetic operations of mutation and crossover. Depending on the problem complexity these steps are repeated $200-2000$ times. The algorithm uses a number of control parameters, and highly elaborated entropy-based measure, $H(N)$ to quantify affinity among members of a population of N lymphocytes. Interestingly, $H(2)$ reduces to the normalized, with respect to the strings length, Hamming distance between these two strings.

A version of this algorithm designed to finding the optimal value of a function over a given domain was proposed by Chun, Jung and Hahn [26]. The algorithm was tested on five cases: the sphere model, step function, five-dimensional Rosenbrock function, *sinc* function, and surface permanent magnet synchronous motor (SPMSM) design. Its quality was compared to the performance of $(1+1)$ evolution strategy and to the standard GA. The authors conclude that their algorithm performs very well for problems with complex fitness landscapes, like *sinc* or multi-modal functions (*e.g.* SPMSM), but in case of smooth shaped functions, evolution strategies are recommended.

In general, immune algorithms with GAs spend a long time searching efficient solutions. To reduce the time complexity, Kawana *et al.* [27], proposed an evolutionary strategy with niche method (ESNM for brevity) as a component of their immune algorithm. The candidates generated by the ESNM are treated as antibodies by the immune system. In each cycle only one tentative solution together with its

Hamming-neighbors is returned, and new antibodies are loaded to the system. The algorithm was applied to solving a set of three nonlinear simultaneous equations, and all the four solutions were extracted faster than by a "standard" immune algorithm.

De Castro and von Zuben [28] proposed much simpler algorithm employing the clonal selection metaphor. It consists of six steps and starts with generating a set of candidate solutions. In the second step a number of best (with respect to an affinity measure) candidates is selected. These candidates are cloned with the rate proportional to their "goodness". Next, each clone is subjected to hypermutation with the rate inversely proportional to its goodness. The most fitted mutants are used to compose the memory set. Finally, a number of poorly behaving antibodies (*i.e.* these with lower fitness) is replaced by randomly generated individuals. In comparison with the algorithm of Fukuda *et al.* the number of control parameters is highly reduced, and the affinity measure (inversely proportional to Euclidean distance) is the only measure employed. The efficiency of the algorithm was tested on three problems: a binary characters recognition problem, a multi-modal function optimization, and an instance of the Traveling Salesmen Problem with 31 cities. Slight modifications of the algorithm result in a useful tool for exploratory data analysis [16]. Thus, an interesting bridge between numerical optimization and data classification, a complex optimization problem, has been established.

Gaspar and Collard [3], proposed another algorithm devoted to time dependent problems. Their implementation distinguishes four stages. The algorithm starts with an initial random population of antigens. Then evaluation phase starts; it relies upon computing so-called exogenic activation (*i.e.* antibody affinity to a given pathogen) and endogenic activation (*i.e.* antibody similarity to other antibodies from the current repertoire). The third stage is clonal selection performed differently on endo- and exo-activated antibodies. The last stage, called recruitment phase, reselects simply antibodies to the next iteration. The experiments reported in [3] show that the system is able to discover new optima, and what is more important, it is able to react over time. Its convergence is never complete, which guarantees its readiness to discover new optima.

4. A new algorithm

The algorithm presented below integrates implicitly many of the mechanisms just described. It is designed to identify as many as possible local optima of a function of n variables over the set $X \subseteq R^n$, where X is a Cartesian product of the intervals $[x_{i,min}, x_{i,max}]$, $i=1,\dots,n$. It uses explicit, *i.e.* real-valued, representation of candidate solutions. To give a conceptual framework for the algorithm we start with an adaptation of the DA to this new representation.

4.1. The diversity algorithm revisited

The main mechanism used by the immune system to attain sufficient diversity is the clonal selection, relying upon producing mutated clones of selected, effective, antibodies. Crossover is not used (at least explicitly) during the system evolution. Further, as discussed in Section 1, mutation can provide higher levels of disruption and exploration than crossover. Replacing binary representation of the antibodies by real-valued representation results – from a formal point of view – in replacing GA component by the ES component that is responsible for the evolution of the mutated clones.

The DA used by Forrest *et al.* can be summarized as follows: (1) present pathogens to the system, (2) find the antibodies with highest affinity to those pathogens, (3) evolve the population of the antibodies. Since the evolution was realized by the GA component, before step (3) it was necessary to assign fitness value to each of the antibodies. This additional step prepares information necessary for the selection procedure of the GA, and it is superfluous in the ES framework.

Another difference with the DA formalism is that when solving optimization problems, the set of pathogens is specified implicitly only contrary to its explicit form of discrete set of the pathogens used in the original DA. Hence the procedure for evolving initial set of antibodies can be imagined as follows:

(1) Construct a finite set of antibodies, Ab.
(2) For each antibody $a \in Ab$ produce a number of mutated clones. Usually this number is proportional to current fitness of a. Denote C the set of produced clones.
(3) For each antibody $a \in Ab$ find a clone $c^a \in C$ with highest affinity to that antibody.
(4) For each pair (a, c^a) check the degree of stimulation of a and c^a. Strongly stimulated antibody is remembered in Ab, while weakly stimulated antibody dies.
(5) Replace a number of weakly stimulated antibodies from Ab by fresh antibodies.
(6) Repeat steps (2)–(5) till a termination condition will be satisfied.

Note that we slightly departed from the original DA. Namely, in step (2) the whole population (instead of a random sample of size σ) of antibodies is mutated. Further, the antibodies are mutated first, and then the mutants are selected for next iterations. This exchange follows from replacing GA formalism by the ES formalism. Step (5) is a further modification of the DA. It accelerates the convergence of the algorithm.

The steps (1)–(6) form a frame algorithm that can be instantiated in different ways dependently on the problem. This algorithm is almost "classical" idiotypic network procedure as described by Farmer, Packard and Perelson in [29], and applied successfully *e.g.* by de Castro and von Zuben in [16]. Another, slightly modified, variant of this procedure was used for exploratory data analysis by Timmis in [15]. Although both authors exploit Jerne's hypothesis, they implement it quite differently. Timmis applies the procedure to construct an immune network resembling idotypic network and representing complex relationships among data, while de Castro and von Zuben try to extract the most relevant patterns in the data, which results in a specific data compression procedure. This last approach seems to be more economic in solving numerical optimization problems.

Some of the steps (2)–(5) require additional comments and specialization.

In step (2) we resigned from choosing a sample of antibodies from the population Ab. This follows from the problem's specifity: the aim is to fit the algorithm to the optimization problem. With fixed populations of pathogens it is possible to use original setting of the DA. When the pathogens are known only implicitly, such a procedure becomes ineffective, however.

The immune system with binary representation produces mutants in an obvious way by using a mutation operator. With real-valued representation we can produce mutants by adding to a given antibody a random number from some interval. Perhaps the simplest way to produce a mutated clone $c(a) = (y_1,\dots,y_n)$ of an antibody $a = (x_1,\dots,x_n)$ is to use the rule:

$$y_i = x_i + \Delta_i \cdot \alpha_t \cdot \max[0,1), \quad i = 1,\dots,n,$$

where α_t is a factor narrowing the search domain (t is the number of current iteration), $\max[0,1)$ represents a uniformly distributed random variable that ranges from zero to one, and:

$$\Delta_i = \begin{cases} x_{i,min} - x_i & \text{if } \max[0,1) < 0.5 \\ x_{i,max} - x_i & \text{if } \max[0,1) \geq 0.5 \end{cases},$$

i.e. the decision which value Δ_i to choose is made randomly. This way each y_i is correctly located within a corresponding domain $[x_{i,min}, x_{i,max}]$. Further, as the number of iterations grows, α_t decreases narrowing the search space. An advantage of the uniform mutation over Gaussian mutation is that this last operator generates mutants located nearby original locations, while the uniform mutants are located far away.

Step (3) is in agreement with the immune network theory [29], and results in the immune memory creation (according to Jerne's hypothesis of course). Recall that antibodies recognize not only pathogens, but they recognize also other antibodies. To keep the population of antibodies of fixed size we use "best-match" strategy similar to that used in the DA. That is, we search for a clone c^a recognizing a given antibody a in the highest degree (this degree is inversely proportional to the Euclidean distance $d(a,c)$), and next we test which element of the pair (a,c^a) copes better with the pathogen. That is we compute the values $f(a)$ and $f(c^a)$, where f stands for the optimized function, and we choose appropriate element.

Step (5) implements the second mechanism used by the immune system to attain high diversity, and it can be implemented in a number of ways. The simplest is to kill a number of worst stimulated antibodies from the repertoire Ab. A more elaborated method is to introduce the mechanism of "network suppression". This prevents convergence of the repertoire to a single antibody, and it is of use when solving multi-modal optimization problems. From a formal standpoint Step (5) resembles the idea of crowding introduced originally by DeJong (consult [2] for details). Empirical studies prove that crowding is a useful technique for maintaining different subpopulation, when an algorithm searches for multiple peaks in a fitness landscape. However, the crowding mechanism alone is not able to identify all the peaks (consult [2] again for numerical details). Below we present numerical results illustrating the efficiency of the algorithm.

4.2. Finding the global optimum over a given domain

From the immune perspective the problem of finding the global optimum over a domain X is equivalent to evolving the antibodies population to identify a single pathogen. To test the algorithm ability in more serious problems, two other functions were used:

– a modified 10-dimensional Grievank function

$$G(x_1,\ldots,x_{10}) = \sum_{i=1}^{10} x_i^2/4000 - 1 + \prod_{i=1}^{10} \cos(x_i/\sqrt{i}), \quad x_i \in [-600,600],\ i = 1,\ldots,10,$$

– 20-dimensional Rastrigin function

$$R(x_1,\ldots,x_{20}) = 200 + \sum_{i=1}^{20} \left(x_i^2 - 10\cos(2\pi x_i)\right), \quad x_i \in [-5.12,5.12],\ i = 1,\ldots,20.$$

These functions are known to be very difficult global optimization test functions. The Grievank function contains about 500 local minima in $[-600,600]^{10}$, which are concentrated around the global minimum located at the origin. Similarly, the Rastrigin function admits a large number of local minima located approximately on the integer coordinates, and the global minimum is at the origin.

For this case, of finding global optimum, the algorithm from Section 4.1 is extremely simple. In the experiments reported below the next parameters were used: memory size, $|Ab| = 20$, ten best antibodies produced mutants (the best antibody produced 20 mutants and the remaining 9 antibodies produced 10 mutants each), five worst antibodies were replaced with new, randomly generated, individuals. Steps (2)–(5) of the algorithm were repeated 400 times. A single run of such an algorithm is referred to as *epoch*. Table 1 presents the results that have been obtained.

Table 1. Results of the experiments averaged over 30 epochs

Optimized function	Best solution	Average solution
Grievank	0.00378	0.0416
Rastrigin	0.00646	0.0665

To compare these results we refer to Hart [30] who studied various genetic algorithm (GA) hybrids equipped with additional local search (LS) optimizers, namely:

(a) – GA + Solis-Wets method,

(b) – GA + conjugate gradient.

These LS optimizers were activated with different probabilities: 0.0625, 0.25, and 1.0 (*i.e.* at each iteration). His results are reported in Table 2. Additionally, Hart analyzed the effect of the elitist and non-elitist (called here *normal*) sampling strategies. Comparing Tables 1 and 2 we see that the immune algorithm is much better than the GA, and only GA-CG method exceeds its quality for Grievank function.

Figure 1 compares the convergence of the immune algorithm with an algorithm without apoptosis realized in step (5). This shows that replacing worst antibodies by fresh individuals improves substantially the convergence. Another interesting property of the algorithm is its sensitivity to the rate of the α_t parameter decrease. In the experiments it was assumed that the parameter decreases geometrically after each T iterations, *i.e.* $\alpha_t = (r)^{t/T} \cdot \alpha_t$ if $(t \bmod T) = 0$. Figure 2 shows the influence of a on the immune algorithm convergence. More details on the new algorithm can be found in the book [8].

Table 2. (quoted after [3], p. 60) Average performance of Gas and GA-LS hybrids with and without elitism. The numbers after corresponding symbols denote the probability of using a LS optimizer

Method	Rastrigin		Grievank	
	Normal	Elitst	Normal	Elitist
GA	166.3	3.6	0.26	0.07
(a) 0.0625	102.9	18.4	0.22	0.11
(a) 0.25	86.8	37.9	0.09	0.04
(a) 1.0	75.3	50.0	0.13	0.08
(b) 0.0625	103.8	24.2	0.02	$8.6 \cdot 10^{-4}$
(b) 0.25	117.8	40.8	$1.2 \cdot 10^{-9}$	$1.0 \cdot 10^{-19}$
(b) 1.0	108.1	59.0	$1.2 \cdot 10^{-19}$	$1.1 \cdot 10^{-19}$

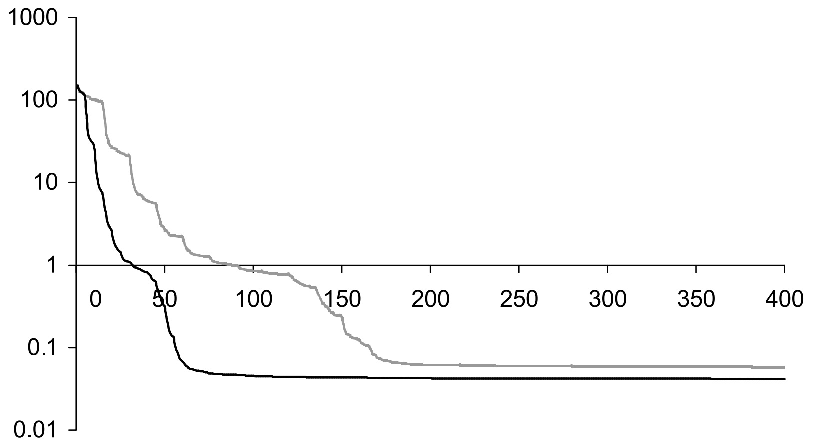

Figure 1. Convergence of the immune algorithm for Grievank function. Gray line presents the convergence of an algorithm consisting of steps (2)–(4) only

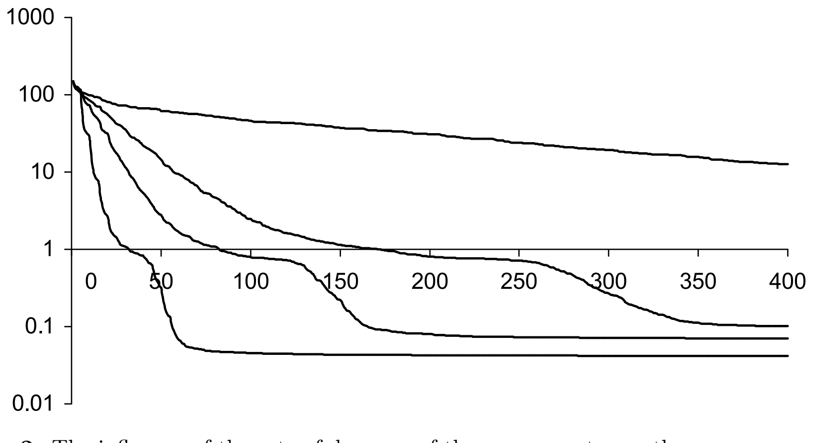

Figure 2. The influence of the rate of decrease of the α_t parameter on the convergence of the immune algorithm. The lower the rate the higher the convergence; the rate varies from 0.99 (upper plot) through 0.9, 0.8 up to 0.5 (lower plot)

4.3. Optimization of multi-modal functions

Careful analysis of the algorithm tuned to optimize multi-modal functions can be found in chapter 9 of [8]. This time, the algorithm is extended by step

(4a) Suppression. For each candidate $m \in M$ if there exists another candidate $m*$ sufficiently similar to m, leave the better candidate in M and replace the worse candidate by a new, randomly generated candidate.

It freely uses the idea of crowding introduced originally by DeJong (consult [2] for details). The simplest way to model "sufficient similarity" is to assume that two candidates m and m' are similar if their distance is less than a prespecified threshold d_{sim}.

While step (3) of the algorithm controls exploitative process, step (4a) is responsible for the explorative properties of the algorithm.

The usefulness of the algorithm was proved by a series of experiments concerned with finding local optima of different one- and two-dimensional functions. Results of these experiments can be found in chapter 9 of [8].

4.4. Non-stationary functions optimization

The task of non-stationary functions optimization is the identification of a series of optima that change their location (and possibly their height) in time. Since each optimum is located in different point of the search space, the algorithm designed to cope with this task can be viewed as pattern tracking algorithm when the solutions are represented by binary strings of fixed length (as in the standard GA). More formally we want to identify all the optima of a function $f(\mathbf{x},t)$ where $\mathbf{x} \in D \subset \mathbf{R}^n$ and t represents time. Typically the domain D is the Cartesian product of the intervals $[x_{i,min}, x_{i,max}]$, $i = 1,\dots,n$.

Evolutionary algorithms designed to cope with such stated task exploit one of the following strategies [31]: the expansion of the memory in order to build up a repertoire of ready responses for environmental changes, or the application of some mechanism for increasing population diversity in order to compensate for changes encountered in the environment. In this last case commonly used mechanisms are: random immigrants mechanism, triggered hypermutation or simply increasing the mutation rate within a standard GA to a constant high level.

The first immune algorithm, Simple Artificial Immune System or Sais [3], was brieftly described in Section 3.2. To be compatible with this approach we show a specialization of the frame algorithm from Section 4.1 to the binary immune optimizer in which the set *Ab* consists of a number of binary strings of fixed length. Its pseudocode is given below:

1. *Fitness evaluation.* For each individual or antibody a in the population *Ab* compute its fitness *i.e.* the value of the objective function $f_p = f(p,t_0)$.
2. *Clonal selection.* Choose n antibodies with the highest affinity to the antigen.
3. *Somatic hypermutation.* Make c_i mutated clones of i^{th} antibody. The clone $c_{(i)}$ with highest fitness replaces original antibody if $f_{c(i)} > f_i$.
4. *Apoptosis.* Each $t_d \geq 1$ iterations replace d weakest antibodies by randomly generated binary strings.

Step 2 of this algorithm can be realized at two different ways. Choosing antibodies for cloning we can use their genotypic or phenotypic affinity. However, it was observed in [32] that the genotypic affinity controls the algorithm behavior more efficiently. This affinity is computed as follows. Suppose $i*$ is an individual with the highest fitness value. Call this individual *current antigen*. Now we compare point-wisely (separately on each segment corresponding to different dimension) current antigen with i^{th} antibody. If the two strings agree on j^{th} position the affinity is increased by 2^{L-j-1}. Figure 3 illustrates exemplary computation of the affinity under the assumption that the binary strings representing 2-dimensional real vectors.

antigen i^*:	0	0	1	1	0	1	0	0	1	1
antibody i:	0	0	0	1	0	0	0	1	1	1
weight	2^4	2^3	—	2^1	1	—	2^3	—	2^1	1
aff(i,i^*)	(16+8+2+1)+(8+2+1)=38									

Figure 3. Computing the affinity between antigen and an antibody

The algorithm prettily behaves in the pattern matching task. For instance, Figure 4 presents its behaviour when a single optimum, or peak, visits periodically three randomly chosen positions in a two dimensional rectangle D. The peak stays at a fixed position through 10 iterations and later it moves to a next position. Under such a setting we can observe the emergence of the immune memory: as time goes by, subsequent positions of the peak are identified faster at each new cycle.

Applications of this algorithm to more serious environments are subject of our intensive studies; consult [33] and the URL: www.ipipan.waw.pl//˜stw/ais.

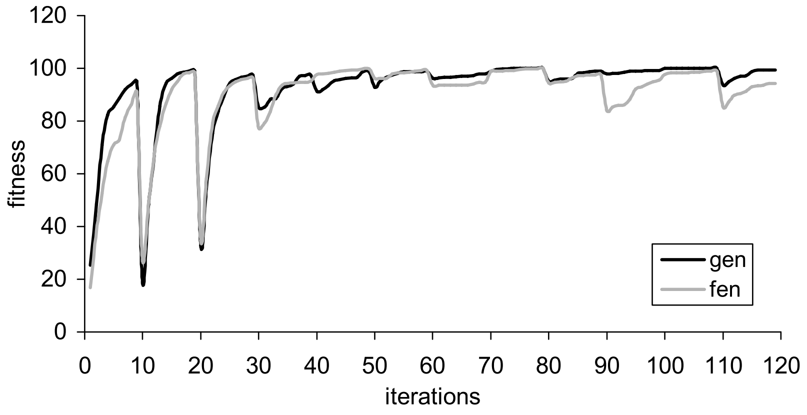

Figure 4. Identification of a periodic non-stationary function. The peak locates in one of three positions. The effects of phenotypic and genotypic affinity are compared

5. Conclusions

A new algorithm for uni- or multi-modal, as well as for non-stationary functions optimization inspired by the immune metaphor has been described. Current results show that the main immune mechanisms of clonal selection and cell suppression can be successfully applied to engineering problems. Using relatively small population of candidate solutions we are able to solve complicated function optimization problems.

Further works should concentrate on giving the algorithm more degree of freedom. Particularly, we need some guidelines on how to control the parameter α_t introduced in Section 4.2, or how to define the similarity of candidate individuals requested in step (4a) introduced in Section 4.3.

References

[1] Corne D, Dorigo M and Glover F (Eds.) 1999 *New Ideas in Optimization*, McGraw-Hill

[2] Goldberg D E 1989 *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley

[3] Gaspar A and Collard Ph 1999 *Proc. 1999 Congress on Evolutionary Computation*, ????, pp. 1859–1866

[4] Park K and Carter B 1994 *Technical Report* BU-CS-94-010, Computer Science Department, Boston University

[5] Spears W M 1992 *Proc. 2*$^{\text{nd}}$ *Foundations of Genetic Algorithms Workshop* (Whitley D, Ed.), San Mateo, CA: Morgan Kaufmann, pp. 221–237

[6] Perelson A S and Weisbuch G 1977 *Reviews of Modern Physics* **69** 1219

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

[8] Wierzchoń S T 2001 *Artificial Immune Systems. Theory and Applications*, Akademicka Oficyna Wydawnicza EXIT, Warsaw (in Polish)

[9] Forrest S, Hofmeyr S A and Somayaji A ???? *Computer Immunology* ???, ???

[10] Kephart J O 1994 *Artificial Life IV* (Brooks R A and Maes P, Eds.), MIT Press, pp. 131–139

[11] Perelson A S 1989 *Immunological Reviews* **110** 5

[12] Hunt J E and Cooke D E 1996 *J. of Network and Computer Applications* **19** 189

[13] Bersini H and Varela F J 1990 *Proc. 1*$^{\text{st}}$ *Workshop on Parallel Problem Solving from Nature, LNCS*, Springer-Verlag **496** 343

[14] Jerne N J 1984 *Immunological Reviews* **79** 5

[15] Timmis J I 2000 *Artificial immune systems: A novel data analysis technique inspired by the immune network theory*, PhD thesis, Department of Computer Science, University of Wales, Aberystwyth

[16] de Castro L N and von Zuben F J 2001 *Data Mining: A Heuristic Approach* (Abbas H A, Sarker R A and Newton Ch S, Eds.), Idea Group Publishing, USA, pp. ???

[17] Beasley D, Bull D R and Martin R R 1993 *Evolutionary Computation* **1** 101

[18] Spears W M 1994 *Proc. 1994 Evolutionary Programming Conference*, World Scientific, pp. 296–317

[19] Forrest S, Javornik B, Smith R E and Perelson A S 1993 *Evolutionary Computation* **1** 191

[20] Smith R E, Forrest S and Perelson A S 1993 *Evolutionary Compuation* **1** 127

[21] Hajela P and Lee J 1996 *Structural Optimization* **12** 11

[22] Hightower R 1996 *Computational Aspect of Antibody Gene Families*, PhD thesis, University of New Mexico, Albuquerque, New Mexico

[23] Oprea M L 1999 *Antibody Repertoires and Pathogen Recognition: The Role of Germline Diversity and Somatic Hypermutation*, PhD thesis, University of New Mexico, Albuquerque, New Mexico

[24] Hart E and Ross P in [5] pp. 185–202

[25] Fukuda T, Mori K and Tsukiyama M in [5] pp. 210–220

[26] Chun J S, Jung H K, Hahn S Y 1998 *IEEE Trans. on Magnetics* **34** (5) 2972

[27] Kawana M, Hiramatsu T, Miyazaki M and Namba N *Dynamic Control of Immune Algorithm by Using Evolutionary Strategy with Niche Method*, Extended abstract available at URL: http://www.iee.or.jp/honbu/back_number/journal/index_back_number /2001/c_es2001.htm

[28] de Castro L N and von Zuben F J 2000 *The Clonal Selection Algorithm with Engineering Applications*, GECCO'00, pp. 36–37

[29] Farmer J D, Packard N H and Perelson A S 1986 *Physica D* **22** 187

[30] Hart E W 1994 *Adaptive Global Optimization with Local Search*, PhD thesis, University of California, San Diego

[31] Grefenstette J 1992 *Parallel Problem Solving from Nature 2* (Manner R and Manderick B, Eds.), Elsevier, pp. 465–501

[32] Wierzchoń S T 2001 *XII Ogólnopolskie Konwersatorium nt. Sztuczna Inteligencja – Nowe Wyzwania*, SzI-16'2001, Siedlce-Warsaw, Akademia Podlaska PAN, WAT, pp. 97–106 (in Polish)

[33] Trojanowski K and Wierzchoń S T 2002 *Intelligent Information Systems* (Kłopotek M A, Wierzchoń S T and Michalewicz M, Eds.), Physica-Verlag (in print)