A T-Cell Algorithm for Solving Dynamic Optimization Problems

Victoria S. Aragón, Susana C. Esquivel
Laboratorio de Investigación y Desarrollo en Inteligencia Computacional*
Universidad Nacional de San Luis
Ejército de los Andes 950
(5700) San Luis, ARGENTINA
{vsaragon, esquivel}@unsl.edu.ar

Carlos A. Coello Coello†
CINVESTAV-IPN (Evolutionary Computation Group)
Computer Science Department
Av. IPN No. 2508, Col. San Pedro Zacatenco
México D.F. 07300, MÉXICO
ccoello@cs.cinvestav.mx

November 2, 2010

Abstract

In this paper, a metaheuristic inspired on the T-Cell model of the immune system (i.e., an artificial immune system) is introduced. The proposed approach (called DTC, for *Dynamic T-Cell*) is used to solve dynamic optimization problems, and is validated using test problems taken from the specialized literature on dynamic optimization. Results are compared with respect to artificial immune approaches representative of the state-of-the-art in the area. Some statistical analyses are also performed, in order to determine the sensitivity of the proposed approach to its parameters.

Keywords: Artificial immune systems, dynamic optimization, metaheuristics.

1 Introduction

In general, the conditions of an optimization problem change by one of the following reasons (or a combination of both) [16]: 1) The objective function

---

*LIDIC is financed by the Universidad Nacional de San Luis and ANPCyT (Agencia Nacional para promover la Ciencia y Tecnología).

†The third author is also affiliated to the UMI 1375 CNRS at CINVESTAV-IPN.

changes itself, or 2) the constraints change. A change in the objective function appears when the purpose of the problem changes. In this case, conditions which were considered desirable before can become undesirable after a change and viceversa. Changes in the constraints, which modify the feasibility of the solutions, are related to resources and their availability. Changes can be small or big, soft or abrupt, chaotic, etc. When changes are big, abrupt or chaotic, the similarity between the solutions found so far and the new ones can be practically null. Even under these hard environments, population-based approaches offer advantages, which are absent in other heuristics when searching for solutions to non-stationary problems. The main advantage relies on the fact that population-based search keeps a set of solutions. Consequently, facing the change, such a population allows the heuristic to move from a solution to another one, in order to determine if any of them has enough merit to continue the search from it, instead of doing it from scratch [5].

Another type of approach are the so-called multipopulation heuristic algorithms, which have been found to be particularly suitable for solving dynamic optimization problems (see for example [13]). Multipopulation approaches are based on the idea of simultaneously tracking a set of local optima. This way, the probability that the new global optimum after the change belongs to the group of those already tracked by the algorithm is higher. The strategy adopted by multipopulation approaches is to split the population into as many sub-populations as possible, so that as many local optima as possible, can be tracked at the same time. Of course, the sub-populations cannot be too small since minimum search capabilities of a single sub-population are required for them to achieve their goal (i.e., to track a local optimum). Individually, each of the sub-populations represents an average optimizer, which, however, should be good enough to follow its local optimum from change to change. During the entire search process, each sub-population controls its local optimum hoping that it could eventually become the global optimum. Thus, it can be said, that the effectiveness of a multipopulation approach comes more from successful gambling than from a flexible (and well-coordinated) optimization process. In any case, as indicated before, this sort of approach has become relatively popular for solving dynamic optimization problems, particularly when using a heuristic known as particle swarm optimization [12] as the main search engine (see [13, 3, 27]).

In recent years, a bio-inspired metaheuristic known as the "artificial immune system" (AIS) has gained popularity in a wide variety of tasks [19, 32, 2, 10, 31]. The AIS is inspired on natural immune systems, which have a number of very interesting features, from a computational point of view, that make them very good candidates to be modelled in a computer. For example, natural immune systems are distributed systems, they are fault-tolerant, they have memory, they are able to distinguish between their own components and those which are foreign, and they learn by experience.

A few AISs have been used before for solving dynamic optimization problems (see for example [30, 9, 25]), but the work in this area is still scarce and there is plenty of room to innovate. Taking this into account, in this paper, the use of a new AIS for solving dynamic optimization problems is introduced, aiming to improve the results obtained by previously proposed AISs for this type of problems.

The remainder of the paper is organized as follows. Section 2 defines what is a dynamic problem. In Sections 3 and 4, the benchmarks used to validate the approach are described. Section 5 describes the AISs that have been previously proposed to solve dynamic optimization problems. In Section 6, the T-Cell Model adopted for this work is described, together with its application to the solution of dynamic optimization problems. In Section 7, an efficiency measure proposed to assess performance of an algorithm adopted to solve dynamic optimization problems is introduced. In Section 8, the experimental setup adopted and the results obtained are discussed. Such results are also compared with respect to other AISs that have been proposed before for dynamic optimization. Additionally, a statistical analysis of results is also presented. Finally, in Section 9, the general conclusions of the paper and some possible paths for future research are provided.

2 Dynamic Problem or Environment

Any time dependent problem can be considered as a dynamic problem. However, from a practical point of view not all dynamic problems cause the same interest. The problems treated in this paper are those where the fitness landscape shows similarities before and after a change occurs. If the problem completely changes, without reference to history, then there is only a sequence of independent problems that have to be solved from scratch. In other words, nothing from the previous search can be used to find the next optimum.

A dynamic fitness landscape or environment is a search space, in which the topological features of the peaks or cones could change over time. Dynamic fitness landscapes present a challenge to any search technique due to the fact that the total or partial number of topological features or problem constraints, or both, could change. So, new cones could appear, and these should be located quickly in order to avoid the loss of potentially good solutions. Consequently, an algorithm working on these landscapes must be able to adapt quickly, and locate and keep the current and the new optimal solutions that arise. For the changes, there may be a variety of dynamic properties. For example, the magnitude of a change could be small, large or chaotic and the speed of a change can be rapid or slow.

Despite the advantages of population-based algorithms, they eventually converge to an optimum and thereby lose the diversity required for efficiently exploring the search space, as well as their ability to react to a change in the environment when such a change occurs. Thus, these approaches often need additional mechanisms to keep diversity in the population.

2.1 Classification of Dynamic Environments

Several classifications of dynamic environments exist. One of them is based on the regularity of the changes:

1. **random:** a change does not dependent of the previous one, and
2. **non-random and predictable:** changes are deterministic. This class can be subdivided into *Cyclic* (after a constant time, the optimum is represented again by the same solution) or *Acyclic* (changes are non-periodic).

The environments, also, can be classified taking into account if the changes are continuous or discrete. If changes are continuous in time and space, then the environment changes little every time one measures it and the optimum is moved from a location to another one, which is sufficiently close from the original one, as to allow that the optimum can be found by performing local search. Now, if the changes are discrete in time and space, they show up and the environment remains stable for a certain period of time. Then, the location of the optimum is modified in such a way that the use of local search is not sufficient to find it [23].

Two test-case generators were selected to validate the approach presented in this paper: STCG [28] and MPB [4]. The reasons for adopting them are as follows. On the one hand, with STCG one can change, cyclically or acyclically, the heights of some peaks from the landscape while the locations of them are fixed. On the other hand, MPB allows to change, in a random way, the locations, heights and slopes from a set of peaks or cones, when these changes are discrete. Thus, these test-case generators provide sufficient flexibility to assess different aspects of the algorithm introduced in this paper. Additionally, these test-case generators have also been used by other authors, which facilitates the comparison of results with respect to other approaches.

3 Simple Test-Case Generator (STCG)

Trojanowski et al. proposed [28] a Simple Test-Case Generator (STCG) for dynamic optimization. Here, the fitness landscape is composed by a set of peaks, such that the best peak heights are modified, cyclically or acyclically, but the location of all the peaks remains fixed. Formally, STCG defines a dynamically changing fitness landscape $f : X \times T \longrightarrow \Re$, where T stands for the (discrete) time, and $X = (x_1, x_2)$ is the set of admissible solutions. The range of the ith variable, $X_i = [lo^i, hi^i]$, is divided into n_i disjoint subintervals $[a^i_j, b^i_j]$, $j = 1, \ldots, n_i$. Thus,

$$A_{ij} = [a^1_i, b^1_i] \times [a^2_j, b^2_j],\ i = 1, \ldots, n_1,\ j = 1, \ldots, n_2, \tag{1}$$

The domain X is decomposed into a family of disjoint subsets A_{ij}, i.e., $X = \cup_i \cup_j A_{ij}$. On each subset A_{ij}, it is defined a unimodal function f_{ij} of paraboloidal shape:

$$f_{ij}(x_1, x_2) = \begin{cases} \alpha(b_i^1 - x_1)(x_1 - a_i^1)(b_j^2 - x_2)(x_2 - a_j^2) & if(x_1, x_2) \in A_{ij} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where $\alpha = 16/[(b_i^1 - a_i^1)^2(b_i^2 - a_i^2)^2]$ is a normalizing constant such that

1. $f_{ij}(x) \in [0, 1]$ for all $x \in A_{ij}$,
2. $f_{ij} = 1$ if x is located in the center of A_{ij},

Then, the value of the fitness function $f(x, t)$ is computed according to the equation:

$$f(x, t) = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} p_{ij}(t) \cdot f_{ij}(x), \tag{3}$$

where $p_{ij} \in [0, p_{max}]$ controls the height of ij^{th} peak in the fitness landscape.

The term $p_{ij}(t)$ is in charge of generating cyclical or acyclical changes on the peak heights. STCG allows to change the height of the peaks localized on: 1) the main diagonal or 2) on both diagonal paths of a chess board. The height of all the remaining peaks is kept unchanged and is kept small in comparison with the height of the varying peaks [28].

4 Moving Peaks Benchmark (MPB)

Branke proposed in [4] the so-called Moving Peaks Benchmark (MPB). Here, the fitness landscape is composed by a set of peaks or cones where location, height and width of each element (peak or cone) can change in a random way. This could cause that the optimum disappears after a change occurs. Formally, MPB defines a dynamically changing fitness landscape $f : X \times T \longrightarrow \Re$, where T stands for the (discrete) time, and $X = (x_1, \ldots, x_5)$ is the set of admissible solutions. Every i^{th} peak or cone, of the landscape, has its height h_i, width w_i, and the coordinates of its maximum $cmax_i$. All the parameters characterizing each peak are generated randomly from the corresponding interval. The fitness function for the i^{th} peak is evaluated as follows:

$$f_i(x_1, \ldots, x_5) = \frac{h_i}{1 + w_i \prod_{j=1}^{5} (x_j - cmax_i[j])^2} \tag{4}$$

while the equation for an i^{th} cone is:

$$f_i(x_1, \ldots, x_5) = h_i - w_i \sqrt{\prod_{j=1}^{5} (x_j - cmax_i[j])^2} \tag{5}$$

Then, the value of the overall fitness function $f(x_1, \ldots, x_5)$ is computed as:

$$f(x_1, \ldots, x_5) = \max\nolimits_{i=1,\ldots,N} f_i(x_1, \ldots, x_5)$$

where N is the number of peaks or cones defined in the landscape [30].

5 Previous Related Work

Several bio-inspired heuristics such as Evolutionary Algorithms or Particle Swarm Optimization, have been used to solve dynamic problems (see for example [13, 3, 8, 27]). However, the use of artificial immune systems to deal with dynamic optimization problems has been relatively scarce. Next, the most representative work found in the specialized literature regarding the use of artificial immune systems to deal with dynamic optimization problems is briefly discussed.

Olivetti de França et al. [21] proposed a multimodal optimization algorithm inspired by the human immune system, which was called OPT-AINET. They encoded the solutions using real number values in an Euclidean shaped space. OPT-AINET is based on the clonal selection principle and it was validated using 18 test functions taken from the specialized literature. In this paper, the authors also modified OPT-AINET in order to make it work in dynamic optimization problems. This version was called DOPT-AINET.

Gaspar and Collard [9] proposed a Simple Artificial Immune System (SAIS). This is an oversimplified model of the immune system designed to be minimal while still capturing the essential mechanisms by which immune systems perform primary and secondary responses to dynamically changing complex environments. SAIS starts with an initially random population of so-called B-Cells, each of which is able to detect a given antigen specified by a binary string. Then, it applies at each generation three operators: Evaluation, Clonal Selection and Recruitment (elimination of undesirable B-cells). SAIS was validated using a pattern tracking problem.

Trojanowski [26] analyzed the efficiency of two mutation operators adopted in a clonal selection based optimization algorithm reported in [24], and called Artificial Immune Iterated Algorithm (AIIA), when used for non-stationary tasks. Both operators use an $\alpha-$stable random number generator. The author argues that appropriate tuning of the α parameter allows to outperform the results of algorithms that use traditional operators. The algorithms were tested with six environments generated with two test-benchmarks: the Simple Test-Case Generator benchmark [28] and the Moving Peaks benchmark [4].

Nanas and De Roeck [18] compared several evolutionary algorithms and artificial immune systems that have been used to solve multimodal dynamic optimization problems. They reviewed several basic evolutionary and immune-inspired approaches available to solve multimodal dynamic optimization problems and identified correspondences and differences among them. They also pointed out the essential computational elements of such approaches.

In another paper, Trojanowski [25] analyzed the efficiency of the B-Cell algorithm when applied to the Moving Peaks benchmark [4]. In this case, the algorithm starts with a randomly generated population of solutions and then performs the process of iterated improvement of the solutions by the repetition of: 1) affinity evaluation and 2) clonal selection and expansion.

Trojanowski et al. [30] compared five types of AISs in dynamic optimization problems: 1) Artificial Immune Iterated Algorithm (AIIA) [29], 2) B-Cell

Algorithm (BCA) [11], 3) Clonal Selection Algorithm (CLONALG) [20], 4) opt-Ainet algorithm [17], and a Simple Artificial Immune System (SAIS) [9]. All of them implement non-deterministic iterated processes of search and all of them work with a population of solutions called antibodies or B-cells. Recall that the antibodies represent candidate solutions to the problem, i.e., points in an n-dimensional real valued search space. The coordinates of these points are represented by real numbers or can be coded as bitstrings. Each algorithm starts with a population of random solutions which are iteratively improved [30]. These approaches were tested with seven types of mutation operators (M_1 to M_7). M_1 changes each coordinate of the antibody by a uniformly distributed value defined by the range of the decision variables and the distance from the position of the point in the search space to the boundary of the space [30]. M_2 uses a Gaussian random numbers generator which is also controlled by the mutation range parameter, but in this case, the new solution candidates can be located in the entire search space [30]. The operator M_3 operates on single elements of the floating point representation, and consists of several steps. In the first step, the operator has to select a position and length of the sequence of elements to be mutated in a string s. It randomly selects an element in s which will be a starting point of the sequence of elements to be mutated. Then, the length of the sequence is randomly generated, too. Finally, every element of the sequence is mutated individually [30]. M_4 is very close to the well-known classical mutation operator in which the subsequent coordinates of a solution are modified by independent Gaussian random variates. But, here, instead of the Gaussian random number generator, the authors adopt the α-stable distribution generator. This distribution is controlled by four parameters [30]. The mutation operator M_5 is a modified version of M_4. Here, the range parameter in the generator is not constant but proportional to the exponent of the normalized fitness of the mutated solution [30]. The range parameter in M_6 is defined in a similar way as for M_4, where the range parameter is constant for all solutions during the entire search process. In M_7, the range parameter varies according to the same rules used in M_5, i.e., it is directly proportional to the exponent of the normalized fitness of the mutated solution (see [30] for futher details). These algorithms were tested with six environments generated with two test-benchmarks: the Simple Test-Case Generator [28] and the Moving Peaks Benchmark [4].

6 T Cell Theory

In this paper, an adaptive immune system model based on the immune responses mediated by the T cells is adopted. Originally, this approach was used to solve static optimization problems [1]. The model is called TCELL, and it considers many of the processes that T cells suffer from their origin in the hematopoietic stem cells in the bone marrow until they become memory cells.

T cells belong to a group of white blood cells known as lymphocytes. They play a central role in cell-mediated immunity. They present special receptors on their cell surface called T cell receptors (TCR[1]). All the T cells originate from hematopoietic stem cells in the bone marrow. The hematopoietic progenitor derived from hematopoietic stem cells populate the thymus and expands by cell division to generate a large population of immature thymocytes [22].

Several subsets of the T cells have been discovered, each with a distinct function. Thus, they can be classified in different populations according to the antigen receptor they express. These antigens receptors could be TCR-1 or TCR-2. Additionally, TCR-2 cells express CD4 or CD8.[2]

Also, T cells can be divided into three groups according to their maturation or development level (phylogenies of the T cells [7]): virgin, effector and memory cells. Virgin cells are those which have never been activated (i.e., they have not suffered proliferation or differentiation). At the beginning, these cells do not express CD4 nor CD8. However, later on, they develop and express both marks, CD4 and CD8. Finally, virgin cells mature and express only one mark, either CD4 or CD8. Before these cells release the thymus, they are subject to both positive selection [14] and negative selection [14]. Positive selection guarantees that the only survivors are the cells with TCRs that present a moderate affinity with respect to the self MHC. Negative selection eliminates the cells with TCRs that recognize self components unrelated to the MHC.

Effector cells are a type of cells that express only one mark, CD4 or CD8. They can be activated by co-stimulating signals plus their ability to recognize an antigen [6, 15]. The immune cells interact through the secretion of cytokines.[3] Cytokines allow cellular comunication. Thus, an immune cell c_i influences the activities (proliferation and differentiation) of another cell c_j through the secretion of cytokines, modulating the production and secretion of cytokines by c_j [7]. In order to activate an effector cell, a co-stimulated signal is necessary. Such signal corresponds to the cytokines secreted from another effector cell. The activation of an effector cell implies that it will be replicated and differentiated. Thus, the proliferation process has as its goal to replicate the cells and the differentiation process changes the clones so that they acquire specialized functional properties.

Finally, the memory cells are those that persist into the host even when the infection or danger has been overtaken, so that in the future, they are able to get stimulated by the same or by a similar antigen. Usually, they respond through proliferation and differentiation, faster with a low dosage of antigens than the memory B cells. It is worth noting that, although the effector and memory cells are replicated, they are not subject to somatic hypermutation. For the effector cells, the differentiation process is subject to the cytokines released by another effector cell. In the model adopted in the work reported here, the differentiation

---

[1] TCRs are responsible for recognizing antigens bound to major histocompatibility complex (MHC) molecules.

[2] Lymphocytes express a large number of surface molecules that can be used to mark different cellular populations. CD means *Cluster Denomination* and indicates the group to which lymphocytes belong.

[3] Proteins act as signal transmitters between cells, and also induce growth, differentiation, activation, etc.

process of the memory cells relies on their own cytokines.

The immune response consists of two phases: the first (called *recognizing phase*) involves the processes that suffer only the virgin cells and the second (called *effector phase*) is related to the processes that suffer the effector and memory cells. The *recognizing phase* has to provide some diversity so that the next phase can produce a cell to eliminate the antigen. Meanwhile, the *effector phase* is in charge of doing this job.

6.1 Proposed Algorithm Based on TCELL

DTC (Dynamic T-Cell) is an algorithm inspired on the TCELL model, which is proposed in this paper to solve dynamic optimization problems. DTC operates on four populations, corresponding to the groups in which the T-cells are divided: (1) Virgin Cells (VC), (2) Effector Cells with cluster denomination CD4 (CD4), (3) Effector Cells with cluster denomination CD8 (CD8) and (4) Memory Cells (MC). Each population is composed by a set of T cells whose characteristics are subject to the population to which they belong.

Virgin Cells (VC) do not suffer the activation process. They have to provide diversity. This is reached through the random acquisition of TCR receptors. Virgin cells are represented by: 1) a *TCR* represented by a bitstring using Gray coding (called TCR_b) and 2) a *TCR* represented by a vector of real numbers (called TCR_r).

Into the natural immune system, the positive and negative selections have to remove the potentially harmful cells. Thus, in DTC, positive selection is in charge of eliminating the cells that recognize the antigen with a low matching. On the other hand, negative selection has to eliminate the cells that have a similar TCR, according to a Hamming or an Euclidean distance, depending on whether the TCR is represented by a TCR_b or by a TCR_r.

Effector Cells are composed by: 1) a TCR_b or TCR_r, if they belong to CD4 or CD8, respectively, 2) a proliferation level and 3) a differentiation level. The goal of this type of cell is to explore in a global way the search space. Thus, CD4 explores the search space, taking advantage of the Gray coding properties (there is only one bit of difference between two consecutive numbers), while CD8 uses real numbers representation (big or small jumps).

The goal of the memory cells is to explore the neighborhood of the best found solutions. These cells are represented by the same components that CD8.

In DTC, the TCR identifies the decision variables of the problem, independently of the TCR representation. The proliferation level indicates the number of clones that will be assigned to a cell and the differentiation level indicates the number of bits or decision variables (according to the TCR representation adopted) that will be changed, when the differentiation process is applied.

The activation of an effector cell, called ce_i, implies the random selection of a set of potential activator (or stimulating) cells. The closest cell to ce_i (using Hamming or Euclidean distance), according to the TCR in the set, is chosen to become the stimulating cell, say ce_j. Then, ce_i proliferates and differentiates.

At the beginning, the proliferation level of each stimulated cell, ce_i, is given by a random value within [1, 5],[4] but then, it is determined taking into account the proliferation level of its stimulating cell (ce_j). If the ce_i is better than ce_j, then ce_i keeps its own proliferation level; otherwise, ce_i receives a level which is 10% lower than the level of ce_j.

Memory cells proliferate and differentiate according to their proliferation level (randomly between 1 and the size of MC[5]) and differentiation level (randomly between 1 and the 90% of the number of decision variables[6]), respectively. Both levels are independent from the other memory cells.

Each type of cell has its own differentiation process, which is blind to their representation and population.

Differentiation for CD4: the differentiation level of ce_i is determined by the Hamming distance between the stimulated (ce_i) and stimulating (ce_j) cells. It indicates the number of bits to be changed. Each decision variable and the bit to be inverted are chosen in a random way. The bits change according to a probability $\text{prob}_{diff-CD4}$. The pseudo-code for the proliferation and differentiation of cell ce_i is shown next:

```
for $np = 1$ to Proliferation Level of $ce_i$ do
  $\text{clone}^{np} \leftarrow ce_i$
  for $nd = 1$ to Differentiation Level of $ce_i$ do
    if $\text{prob}_{diff-CD4}$ then
      $k \leftarrow U(1, \mid vd \mid)$
      $l \leftarrow U(1, \mid bits_k \mid)$
      Invert the $l^{th}$-bit of $vd_k$ of the $\text{clone}^{np}$
    end if
  end for
end for
```

where $U(1, w)$ refers to a random number with a uniform distribution in the range (1,w), $\mid vd \mid$ is the number of decision variables of the problem, $\mid bits_k \mid$ is the number of bits to represent the k^{th} decision variable and vd_k indicates the k^{th} decision variable.

Differentiation for CD8: the differentiation level for cell ce_i is related to its stimulating cell (ce_j). If the TCR_r of the ce_j is better than the TCR_r of the stimulated cell ce_i (according to the objective function value), then the level (for ce_i) is a random number within [1, $\mid dv \mid$[7]]; otherwise, it is a random value within [1, $\mid dv \mid /2$], where $\mid dv \mid$ is the number of decision variables of the problem. Each variable to be changed is chosen in a random way and it is modified according to the following equation:

---

[4] This value was derived after numerous experiments.

[5] This is an arbitrary value adopted in order to avoid overloading the number of required parameters.

[6] This value was set thinking on performing an intensive local search.

[7] If the stimulating cell is better, then ce_i should change more decision variables

$$x^{'} = x \pm U(0, lu - ll)^{U(0,1)} \tag{6}$$

where x and $x^{'}$ are the original and the mutated decision variables, respectively. lu and ll are the upper and lower bounds of x, respectively. At the moment of the differentiation of a cell (ce_i), the value of the objective function of its stimulating cell (ce_j) is taken into account. In order to determine if $r = U(0, lu - ll)^{U(0,1)}$, will be added or subtracted to x, the following criteria are considered: if ce_j is better than ce_i and the decision variable value of ce_j is less than the value of ce_i, or if ce_i is better than ce_j and the decision variable value of ce_i is less than the value of ce_j, then r is subtracted from x; otherwise, r is added to x. Both criteria aim to guide the search towards the best solutions found so far. The pseudo-code for the proliferation and differentiation of the cell ce_i with the stimulating cell ce_j is shown next:

```
for $np = 1$ to Proliferation Level of $ce_i$ do
  clone$^{np} \leftarrow ce_i$
  for $nd = 1$ to Differentiation Level of $ce_i$ do
    $k \leftarrow U(1, \mid vd \mid)$
    $r \leftarrow U(0, lu - ll)^{U(0,1)}$
    if $f(ce_{jTCR_r})$ is better than $f(ce_{iTCR_r})$ and $ce_{jTCR_{r_k}} < ce_{iTCR_{r_k}}$
    o $f(ce_{iTCR_r})$ is better than $f(ce_{jTCR_r})$ and $ce_{jTCR_{r_k}} > ce_{iTCR_{r_k}}$
    then
      clon$^{np}TCR_{r_k} \leftarrow ce_{iTCR_{r_k}} - r$
    else if $f(ce_{jTCR_r})$ is better than $f(ce_{iTCR_r})$ and $ce_{jTCR_{r_k}} > ce_{iTCR_{r_k}}$
    o $f(ce_{iTCR_r})$ is better than $f(ce_{jTCR_r})$ and $ce_{jTCR_{r_k}} < ce_{iTCR_{r_k}}$
    then
      clone$^{np}TCR_{r_k} \leftarrow ce_{iTCR_{r_k}} + r$
    else
      add or subtract $r$ with probability 50%
    end if
  end for
end for
```

where $U(w_1, w_2)$ refers to a random number with a uniform distribution in the range (w_1,w_2), $\mid vd \mid$ is the number of decision variables of the problem, lu_x and ll_x are the upper and lower bounds of x, respectively. *iter* indicates the number of iterations until reaching the maximum number of evaluations for a change. $f(ce_{hTCR_r})$ is the objective function value for the TCR$_r$ of the cell ce_h, and $ce_{hTCR_{r_k}}$ indicates the k^{th} decision variable of the cell h.

Differentiation for MC: the number of decision variables to be changed is determined by the differentiation level of the cell to differentiate. Each variable to be changed is chosen in a random way and it is modified according to the following equation:

$$x^{'} = x \pm \left(\frac{U(0, lu_x - ll_x)}{100 iter} \right)^{U(0,1)} \tag{7}$$

where x and $x^{'}$ are the original and the mutated decision variables, respectively. $U(0,w)$ refers to a random number with a uniform distribution in the range (0,w). lu_x and ll_x are the upper and lower bounds of x, respectively. $iter$ indicates the number of iterations until reaching the maximum number of evaluations for a change. In a random way, it is decided if $r = \left(\frac{U(0, lu_x - ll_x)}{100 iter} \right)^{U(0,1)}$ will be added or subtracted to x.

The general structure of the algorithm proposed here for dynamic optimization problems is given in Algorithm 1. Figure 1 shows the transit of cells. Each arrow indicates the direction in which the cells are inserted into each population.

Algorithm 1 DTC Algorithm

```
Initialize_VC();
Evaluate_VC();
Assign_Proliferation();
Divide_CDs();
Positive_Selection_CD4();// eliminate the cells in CD4 with worst objective function value
Positive_Selection_CD8();// eliminate the cells in CD8 with worst objective function value
Negative_Selection_CD4();// eliminate the most similar cells in CD4
Negative_Selection_CD8();// eliminate the most similar cells in CD8
while A predetermined number of change has not been reached do
    while A predetermined number of evaluations has not been performed do
        Active_CD4();
        Sort_CD4();
        Comunication_CD4_CD8();
        Active_CD8();
        Sort_CD8();
        Insert_CDs_en_MC();
        for $i = 1$ to $\text{rep}_{MC}$ do
            Active_MC();
        end for
        Sort_CM();
    end while
    Statistics();
    Change_Function();
    Re-evaluate_Populations();
end while
```

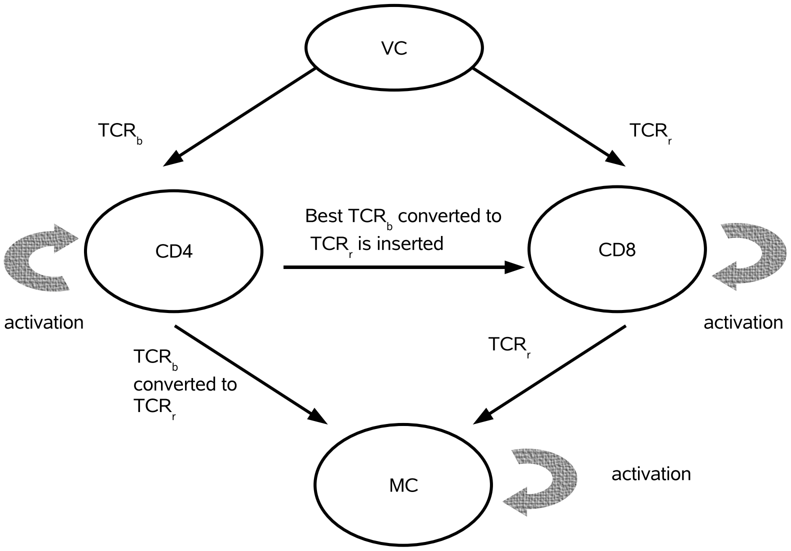


Figure 1: Transit of cells.

The algorithm works in the following way. At the beginning, the TCR$_b$ and TCR$_r$ from the virgin cells are initialized in a random way, according to the TCR's encoding (step 1). Then, each TCR of a virgin cell is evaluated (step 2). In step 3, the proliferation levels are assigned. Then, in step 4, the virgin cells are divided. Virgin cells with the best TCR$_b$ will conform CD4, while virgin cells with the best TCR$_r$ will conform CD8. Each effector cell will inherit the proliferation level of the virgin cell which received the TCR.

The negative and positive selections are applied to each effector population (CD4 and CD8). The first selection eliminates 10% of the worst cells and the second selection eliminates cells that are similar between them (keeping the best from them). This mechanism works in the following way: for each effector cell, one searches inside its population for the closest cell (using Hamming or Euclidean distance according to the TCR cell) and the worst between them is eliminated. This process reduces the effector's population sizes.

The first iteration (step 9) is controlled by the number of changes of the objective function. Besides, for each change, a maximum number of objective function evaluations is allowed (step 10). The steps inside the last iteration are: to activate the CD4 population; in other words, to perform proliferation and differentiation of all the cells from CD4 (step 11). Then, these cells are sorted in a descending way in order to get the best cell (step 12). There exists a comunication process between CD4 and CD8 (step 13), where the best cell from CD4 replaces the worst cell from CD8. Since the representation schemes of the TCR, for CD4 and CD8, are different, before the insertion of the best cell from

CD4 (with TCR_b) into CD8, the receptor has to be converted into a real-values vector (TCR_r). For this process, the following equation is used, which takes as input a bitstring generated with Gray coding and returns a real number (this process is applied as many times as decision variables has the problem):

$$dv_j = l_{lj} + \frac{\sum_{i=0}^{L_j} 2^{L_j - i} dv'_{ij}(lu_j - ll_j)}{2_j^L - 1} \tag{8}$$

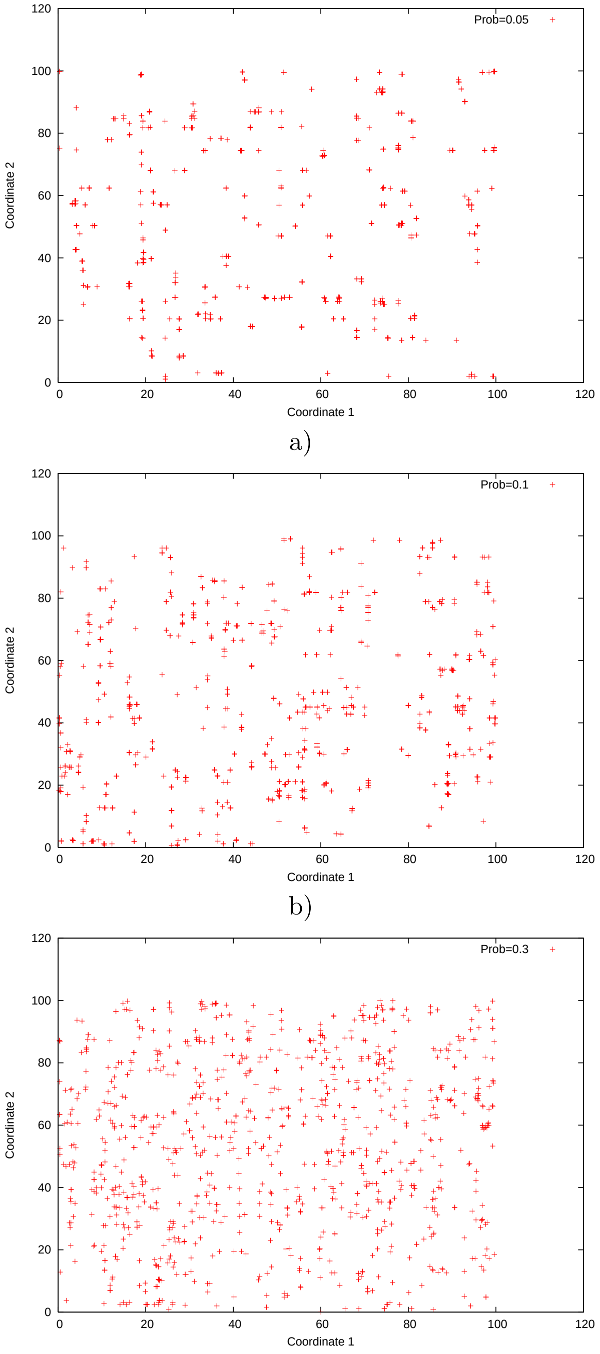
where dv_j is the j^{th} decision variable with $j = 1, \ldots,$ number of decision variables of the problem, L_j is the number of bits for the j^{th} decision variable, lu_j and ll_j are the upper and lower limits for the decision variable dv_j, respectively. And dv'_{ij} is the i^{th} bit of the bitstring that represents dv_j. Also, equation (8) is used when a cell from CD4 has to be decoded in order to be evaluated.

After the above communication process, the CD8 population is activated. This means that proliferation and differentiation of all the cells are performed from CD8 (step 14), which are sorted in descending order (step 15).

The best solutions from CD4 and CD8 are inserted or are used to replace the 50% of the worst solutions in MC (depending on whether or not, MC is empty) (step 16). Again, equation (8) is used to convert the TCR_b from CD4 cells before inserting them, as cells with TCR_r, into MC. Next, the cells from MC are activated a certain (predefined) number of times, rep_MC (step 18).

It is assumed that the algorithm `knows` when the environment has changed (step 23), since that information is required in order to re-evaluate the populations (step 24).

In order to corroborate if the differentiation operators work in the desired way, a bi-dimensional point was generated in the range $[0, 100]$ and the three operators were applied to it in both dimensions, 1000 times. For binary Gray coding (CD4), 40 bits were used to represent each decision variable. The number of bits to be changed was set to 18 (differentiation level). Figure 2 shows the results of the application of the differentiation operator for CD4 with different $\mathrm{prob}_{diff-CD4}$. As the likelihood increases, the operator finds more different solutions. Figure 3 shows the superposition of the generated points for each operator. It can be seen that the distribution of CD8 points is similar to that of the CD4 points with $\mathrm{prob}_{diff-CD4}$=0.3, which implies a good global search capability. Furthermore, it should be clear how the operator for MC performs an in-depth local search.



a)

b)

c)

Figure 2: Distribution of 1000 mutated points using the differentiation operator for CD4 (on the same original point) with: a) prob$_{diff-CD4}$=0.05, b) prob$_{diff-CD4}$=0.1 and c) prob$_{diff-CD4}$=0.3.

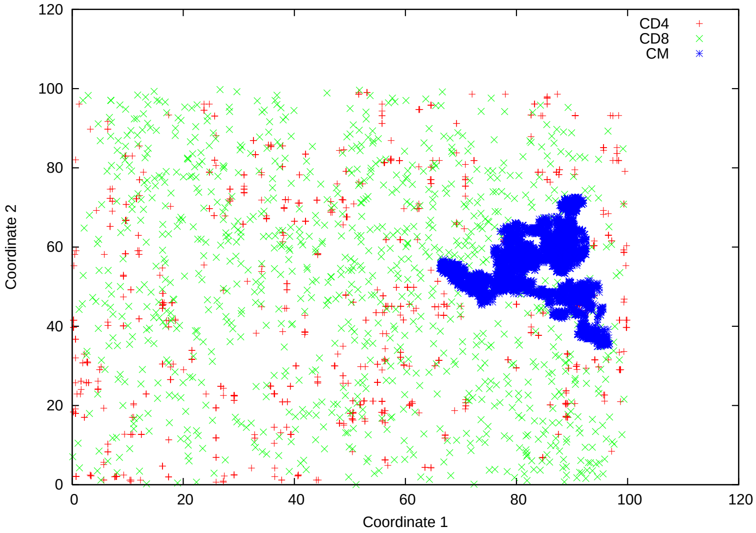


Figure 3: Overlapping of 1000 mutated points generated by the application of the three differentiation operators (CD4 with prob$_{diff-CD4}$=0.1, CD8 and CM).

7 Efficiency Measure

In order to assess the performance of the algorithm, the following measure, which is relatively popular in the literature, was used [30]: Offline error (oe), which represents the average deviation of the best individual evaluated since the last change from the optimum. It is defined by:

$$oe = \frac{1}{N_c}\sum_{j=1}^{N_c}(\frac{1}{N_e(j)}\sum_{i=1}^{N_e(j)}(f_j^* - f_{ji}^*)) \tag{9}$$

where N_c is the total number of fitness landscape changes within an experiment, $N_e(j)$ is the number of solution evaluations performed for the j^{th} state of the landscape, f_j^* is the value of the optimal solution for the j^{th} landscape and f_{ji}^* is the current best fitness value found for the j^{th} landscape [30].

The ideal value for oe is zero, which would mean that the optimum was found in the evaluation of the first solution for each state of the landscape. The first ten oe values (corresponding to changes 1 to 10) from each run are discarded in order to allow the algorithm to reach some stability.

Environment	STCG$_{12nc}$	STCG$_{10c}$	STCG$_{20c}$	STCG$_{20nc}$
Number of peaks	36	100	100	100
Number of moving peaks	12	10	20	20
Number of dimensions	2	2	2	2
Coordinate	[0, 6]	[0, 10]	[0, 10]	[0, 10]
Type of change	non-cyclic	cyclic	cyclic	non-cyclic
p_{max}	100	100	100	100
Change every x evaluations	5000	5000	5000	5000

Table 1: Parameters adopted for the environments generated by STCG.

8 Numerical Experiments

8.1 Dynamic Environments and Their Parameters

To validate the proposed DTC algorithm, six fitness landscapes created with two test functions generators for dynamic environments were adopted. The environments generated by STCG (see Section 3) and their features are the following (Table 1 summarizes the parameters of each environment generated by STCG):

1. STCG$_{12nc}$: this environment has 2 dimensions, and 36 peaks where 12 of them change their heights acyclically.
2. STCG$_{10c}$: this environment has 2 dimensions, and 100 peaks where 10 of them change their heights cyclically.
3. STCG$_{20c}$: this environment has 2 dimensions, and 100 peaks where 20 of them change their heights cyclically.
4. STCG$_{20nc}$: this environment has 2 dimensions, and 100 peaks where 20 of them change their heights acyclically.

The environments generated by MPB (see Section 4) and their features are the following (Table 2 describes the parameters adopted for them):

1. MPB$_5$ (scenario 1): this environment has 5 dimensions, and 5 peaks where all of them change, in a random way, their heights, widths and locations.
2. MPB$_{50}$ (scenario 2): this environment has 5 dimensions, and 50 cones where all of them change, in a random way, their heights, widths and locations.

For MPB$_5$ (scenario 1) and MPB$_{50}$ (scenario 2), the standard settings given in the web page: `http://www.aifb.uni-karlsruhe.de/~jbr/MovPeaks/movpeaks/` were adopted.

Environment	MPB_5 (scenario 1)	MPB_{50} (scenario 2)
movrand	1	1
Number of peaks	5	50
Number of dimensions	5	5
Height	[30, 70]	[30, 70]
Stdheight	50	50
Width	[0.0001, 0.2]	[1.0, 12.0]
Stdwidth	0.1	0.0
Coordinate	[0, 100]	[0, 100]
Height_severity - width_severity - vlength	7.0 - 0.01 -1.0	7.0 - 1.0 - 1.0
Use_basis_function	FALSE	FALSE
Correlation lambda	0.0	0.0
Change every x evaluations	5000	5000
Peak_function	function1	cone
Change_stepsize	constant	constant

Table 2: Parameters adopted for the environments generated by MPB.

8.2 Parameters for DTC

The required parameters for the proposed DTC are the following: size of VC, CD4, CD8 and MC; number of repetitions for the activation of MC (rep_MC) and probability of application of the differentiation operator for CD4 ($prob_{diff-CD4}$).

Binary Gray encoding was adopted (for VC and CD4) with 40 bits for each decision variable. 50 independent runs and 110 changes of the objective function were performed for each environment.

The best compromises of parameter values, for each benchmark were empirically derived and are as follows:

- For the environments generated by STCG: size of VC, CD4 and CD8 of 300 cells, size of MC of 3 cells, $\text{prob}_{diff-CD4} = 0.3$, rep_MC = 10 repetitions,
- For the environments generated by MPB: size of VC, CD4 and CD8 of 300 cells, size of MC of 3 cells, $\text{prob}_{diff-CD4} = 0.1$, rep_MC = 50 repetitions.

Results were compared with respect to those obtained by the best combination of AIS and a mutation operator presented in [30] for each environment, and are as follows:

- For STCG_{12nc}: AIIA - M_7 - $\alpha = 0.5$; CLONALG-M_2; Sais-M_2; BCA - M_5 - $\alpha = 2.0$ and opt-Ainet -M_1.
- For STCG_{10c}: AIIA - M_6 - $\alpha = 1.95$ and $\alpha = 2.0$; CLONALG-M_2; Sais-M_2; BCA M_6 - $\alpha = 1.75$ and M_7 - $\alpha = 1.0$ to $\alpha = 1.90$ and $\alpha = 2.0$; opt-Ainet -M_1.
- For STCG_{20c}: AIIA - M_6 - $\alpha = 2.0$; CLONALG-M_2; Sais-M_2; BCA M_6 - $\alpha = 2.0$ and opt-Ainet -M_1.

- For STCG_{20nc}: AIIA - M_7 - $\alpha = 0.5$; CLONALG-M_2; Sais-M_2; BCA - M_5 - $\alpha = 2.0$ and opt-Ainet -M_1.
- For MPB_5: AIIA - M_6 - $\alpha = 2.0$; CLONALG-M_2; Sais-M_2; BCA - M_5 - $\alpha = 1.75$ and opt-Ainet -M_2.
- For MPB_{50}: AIIA - M_5 - $\alpha = 2.0$; CLONALG-M_3; Sais-M_2; BCA - M_5 - $\alpha = 2.0$ and opt-Ainet -M_1.

These approaches executed 50 independent runs. The problems experienced 110 changes of the fitness landscape and, approximately 5000 objective function evaluations took place per change. For these problems, the authors of the other methods with respect to which results were compared, reported the mean *oe* value.

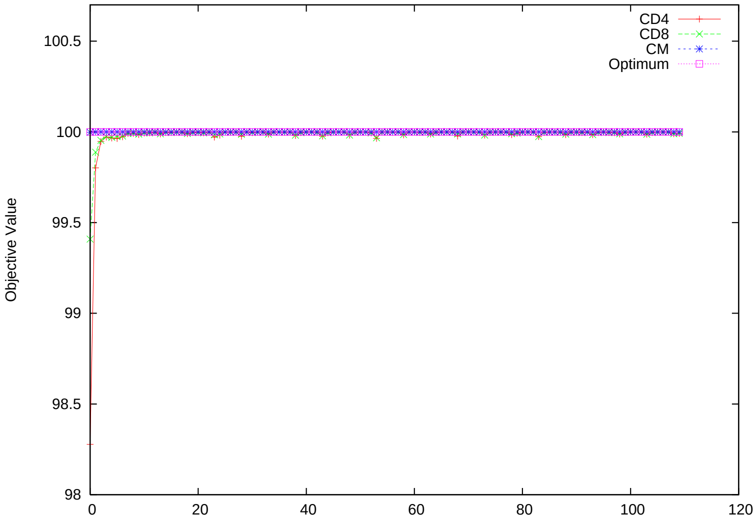
8.3 Analysis of Results

Table 3 shows the best, worst and mean *oe* values obtained by DTC for the environments under study. In Figures 4, 5 and 6, the best solution found in each population is compared, before a change occurs, for each environment. From Figures 4a) and 4b), it can be seen that cyclic changes do not seem to be a challenge for DTC.

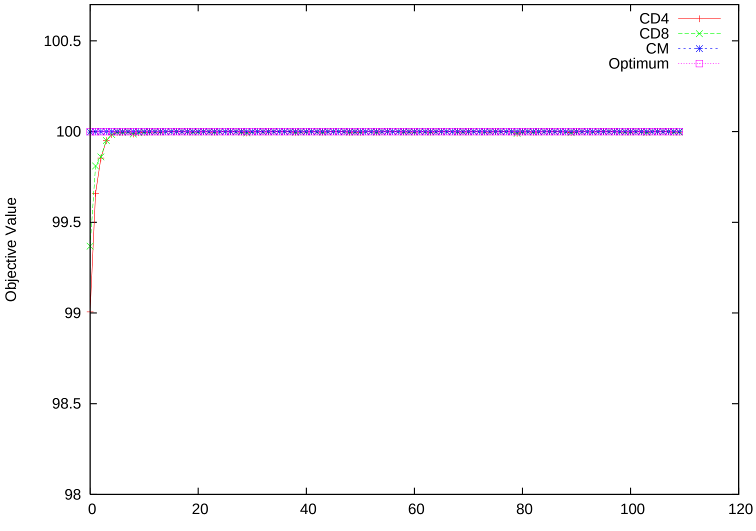
From Figures 5a) and 5b), it can be observed that acyclic changes affect negatively the behavior of DTC, but before a change occurs, DTC can find good solutions. However, its performance deteriorates as the number of changing peaks in the landscape grows.

Environment	Best *oe*	Worst *oe*	Mean *oe*
STCG_{10c}	0.00	0.00	0.00
STCG_{20c}	0.00	0.12	0.03
STCG_{12nc}	0.30	1.1	0.59
STCG_{20nc}	0.48	1.62	0.92
MPB_5	0.81	1.33	1.07
MPB_{50}	1.97	3.45	2.50

Table 3: Values of *oe* obtained by DTC for each environment.

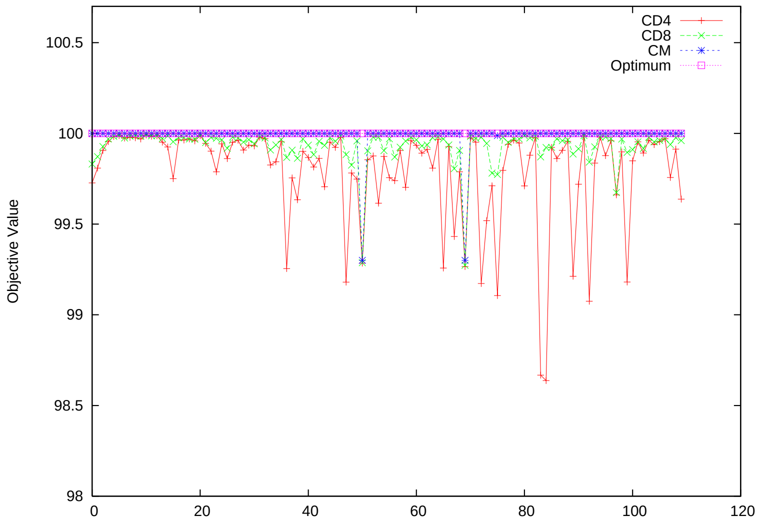


a)

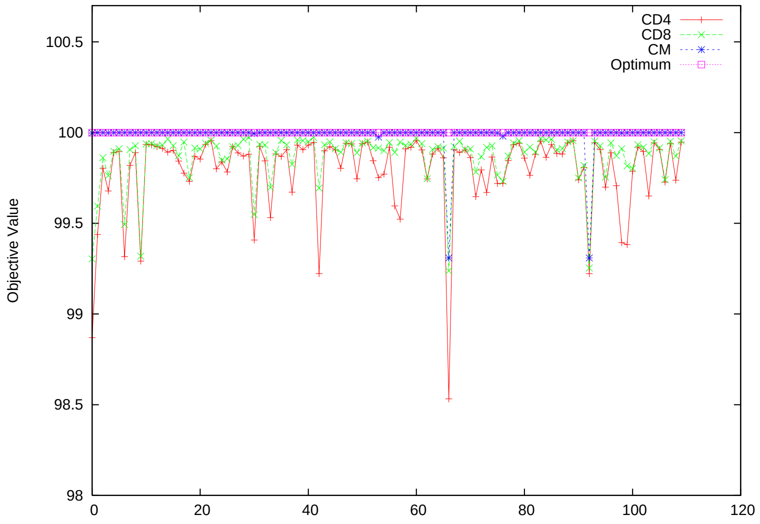


b)

Figure 4: Comparison between the best cell found, before a change, by each population for: a) STCG_{10c}, and b) STCG_{20c}.



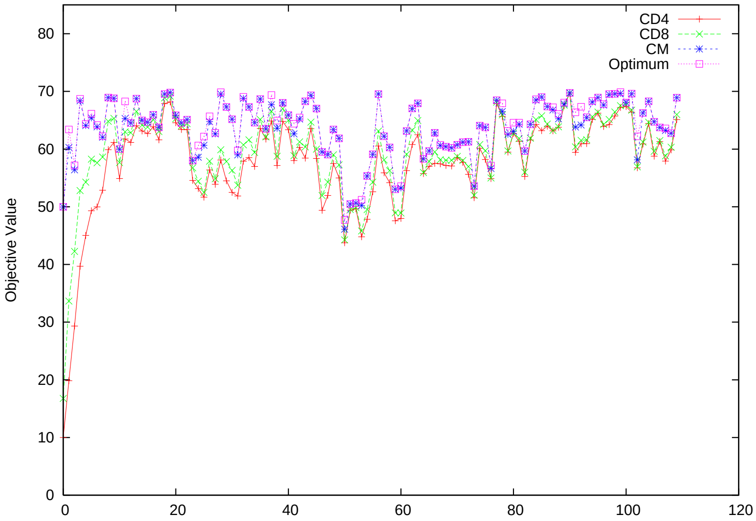
a)



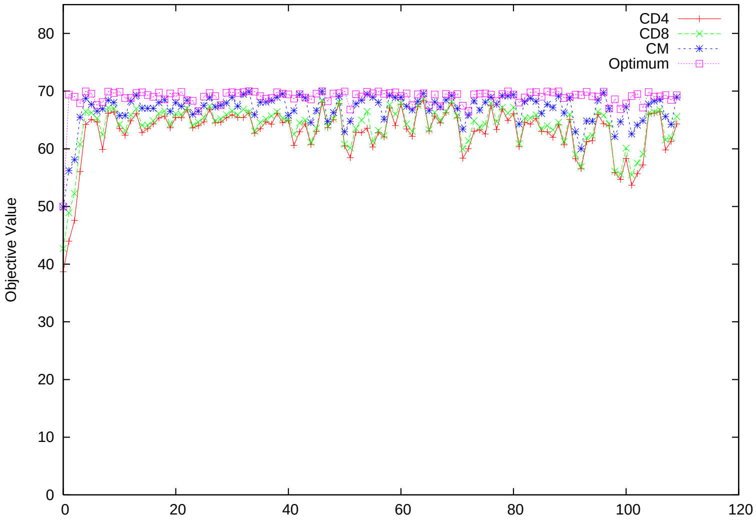
b)

Figure 5: Comparison between the best cell found, before a change, by each population for: a) STCG_{12nc}, and b) STCG_{20nc}.

From Figure 6, it can be seen that the presence of a change has more impact when the landscape has fewer peaks. For MPB_{50} and MPB_5, the solutions found by CD4 seem to be a starting point in order for CD8 and MC to improve them.



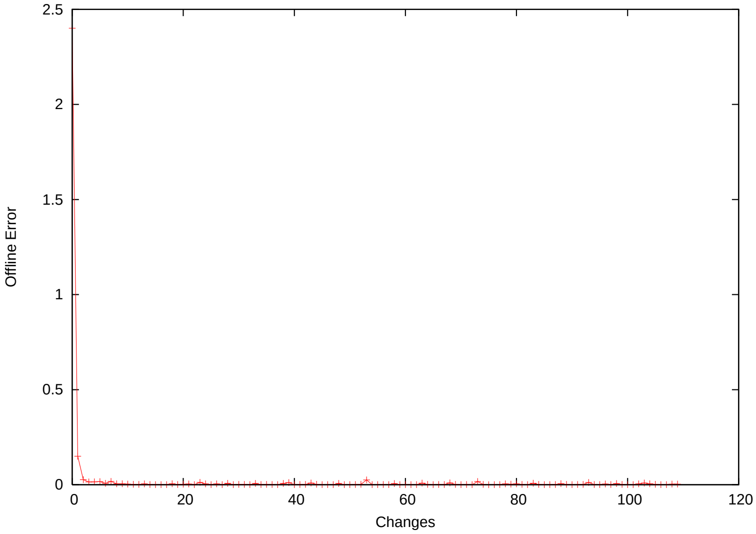
a)



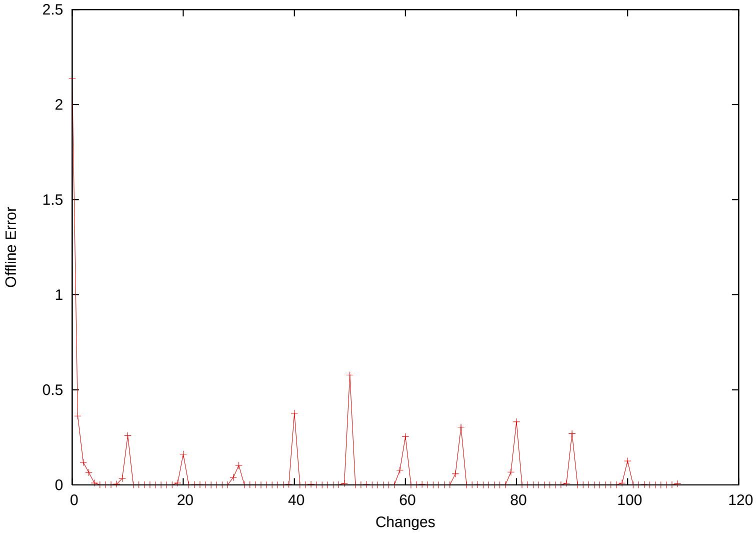
b)

Figure 6: Comparison between the best cell found, before a change, by each population for: a) MPB_5, and b) MPB_{50}.

Figures 7, 8 and 9 support these findings. Here, it can be observed how the offline errors are consistent with the facts presented before.



a)



b)

Figure 7: Offline Error: a) STCG_{10c}, b) STCG_{20c}.

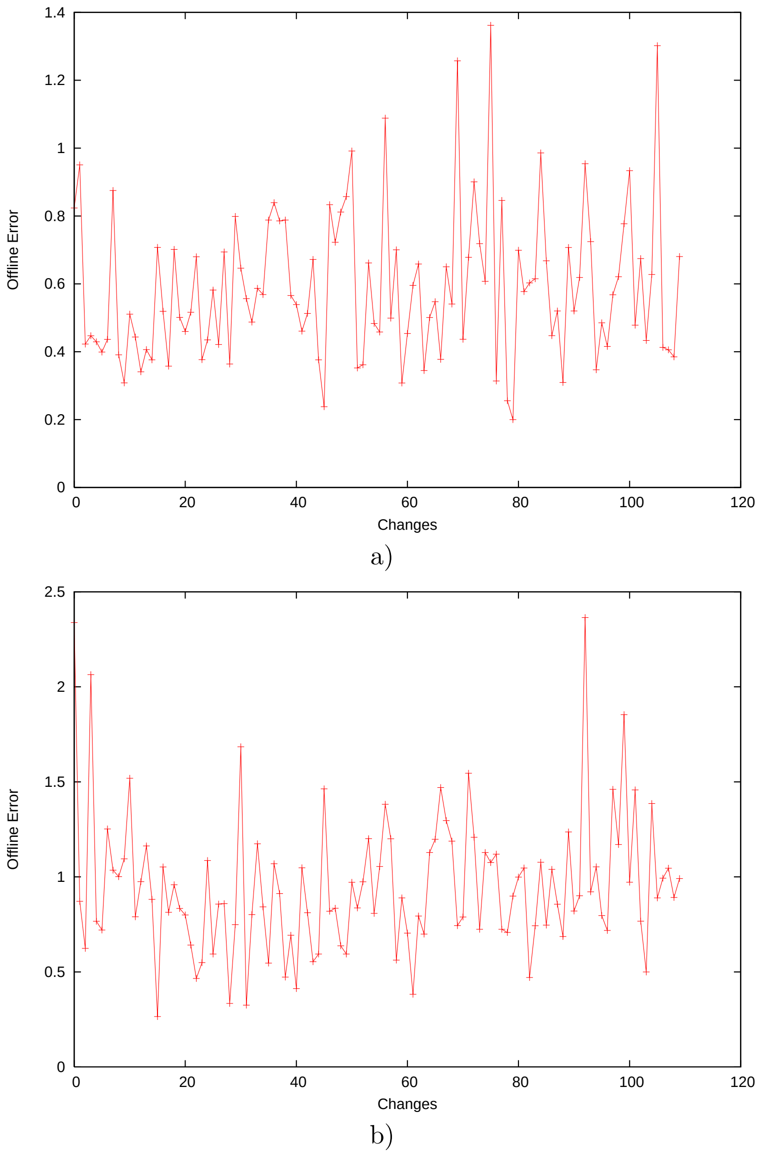
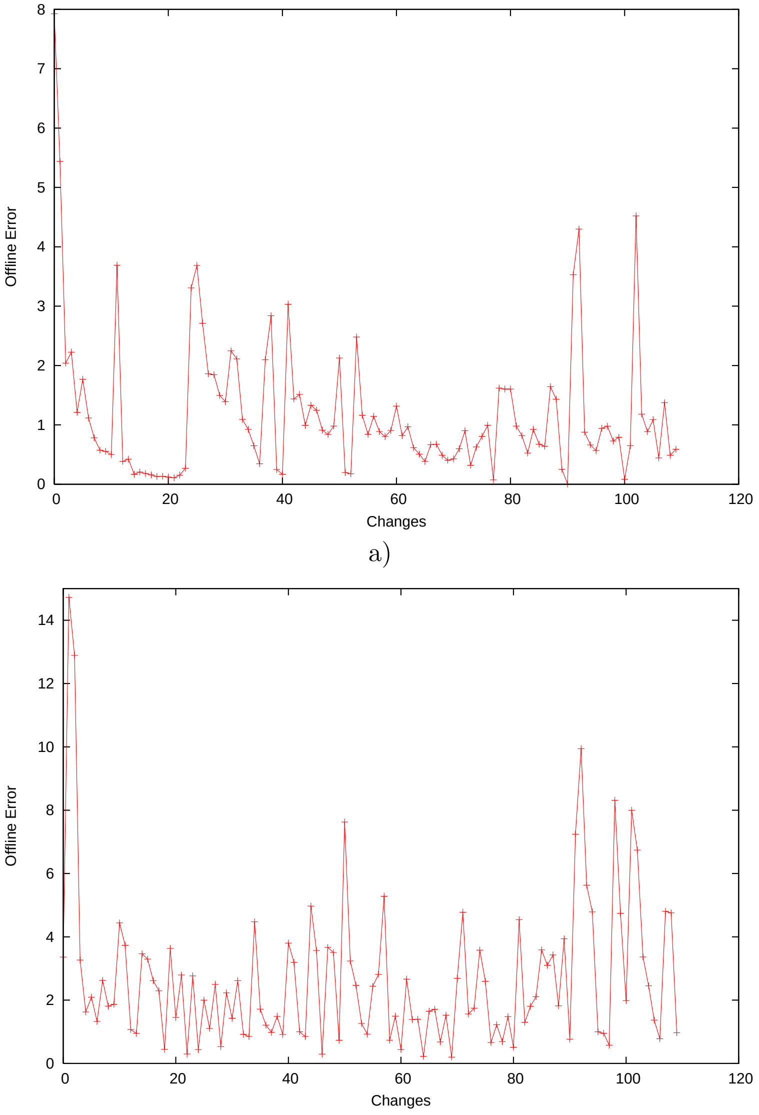


Figure 8: Offline Error: a) STCG$_{12nc}$, b) STCG$_{20nc}$.



a)

b)

Figure 9: Offline Error: a) MPB_5, b) MPB_{50}.

When comparing the proposed DTC with respect to the AIS presented in [30] (see Table 4), it can be seen that DTC obtained better results in all the environments, except for MPB_5. If the approaches are ranked, according to Table 4, the following ranking is obtained (from best to worst performance): DTC, BCA, AIIA, opt-Ainet, CLONALG and Sais. It is worth noting that all these approaches use real numbers representation, except for DTC, which uses

Approach	STCG_{10c}	STCG_{20c}	STCG_{12nc}	STCG_{20nc}	MPB_5	MPB_{50}
AIIA	1.04	1.03	1.08	2.26	0.71	3.46
CLONALG	3.02	3.14	2.54	6.23	11.71	10.53
Sais	3.53	3.56	2.62	6.56	12.16	11.57
BCA	**0.00**	0.18	1.04	1.60	**0.39**	2.69
opt-Ainet	1.40	4.20	1.80	3.32	2.39	4.76
DTC	**0.00**	**0.03**	**0.59**	**0.92**	1.07	**2.50**

Table 4: Mean values of the *oe* obtained by each approach.

a mixed encoding consisting of both, binary and real numbers. Indeed, this could be one of the main reasons for which DTC performs better than the other approaches in the problems adopted. There are, however, other differences between DTC and the other approaches with respect to which results were compared. First, it is worth noting that DTC uses three differentiation operators while the other approaches adopt a single mutation operator. On the other hand, AIIA, CLONALG and Sais, clone and mutate only the best solutions and remove the worst solutions at each iteration. In contrast, DTC only dismisses the worst solutions at the beginning of the search process when positive selection is applied to both CD4 and CD8, and then DTC clones all members of the populations (even when the differentiation level is given by a random number). With respect to BCA, DTC differs from it in that BCA inserts, at each iteration, a randomly generated antibody and DTC never does it. Also, DTC uses two populations to perform global search and a third population whose role is to act as a local search engine. In contrast, BCA is a quasi-multipopulation approach. Finally, opt-Ainet intoduces a randomly generated antibody at each iteration too but, depending on the average fitness, it could, at each iteration, remove the worst of the similar solutions and to insert a number of randomly generated antibodies. However, DTC only removes the worst similar solutions at the beginning of the search, when negative selection is applied to both CD4 and CD8. Thus, DTC is able to keep diversity in the populations without adding any extra mechanisms.

8.4 Statistical Analysis

In order to statistically analyze the effect of each parameter on the behavior of DTC, it was tested with different parameters settings. The settings adopted were empirically derived after numerous experiments.

Since the VC population is initialized only once (it is not involved in the search process, but only acts as the starting point of the search), its size was fixed in 300 cells. The remaining parameters have different levels, which were defined as follows:

- CD4 and CD8 have four levels: 50, 100, 200 and 300 cells;
- MC has two levels: 3 and 10 cells;
- $\mathrm{prob}_{diff-CD4}$ has three levels: 0.05, 0.1 and 0.3;

Parameters Setting ID	VC	CD4	CD8	MC	rep_MC	prob$_{diff-CD4}$
01	300	50	50	3	10	0.05
02	300	50	50	3	10	0.1
03	300	50	50	3	10	0.3
04	300	50	50	10	10	0.05
05	300	50	50	10	10	0.1
06	300	50	50	10	10	0.3
07	300	50	50	3	50	0.05
08	300	50	50	3	50	0.1
09	300	50	50	3	50	0.3
10	300	50	50	10	50	0.05
11	300	50	50	10	50	0.1
12	300	50	50	10	50	0.3
13	300	100	100	3	10	0.05
14	300	100	100	3	10	0.1
15	300	100	100	3	10	0.3
16	300	100	100	10	10	0.05
17	300	100	100	10	10	0.1
18	300	100	100	10	10	0.3
19	300	100	100	3	50	0.05
20	300	100	100	3	50	0.1
21	300	100	100	3	50	0.3
22	300	100	100	10	50	0.05
23	300	100	100	10	50	0.1
24	300	100	100	10	50	0.3
25	300	200	200	3	10	0.05
26	300	200	200	3	10	0.1
27	300	200	200	3	10	0.3
28	300	200	200	10	10	0.05
29	300	200	200	10	10	0.1
30	300	200	200	10	10	0.3
31	300	200	200	3	50	0.05
32	300	200	200	3	50	0.1
33	300	200	200	3	50	0.3
34	300	200	200	10	50	0.05
35	300	200	200	10	50	0.1
36	300	200	200	10	50	0.3
37	300	300	300	3	10	0.05
38	300	300	300	3	10	0.1
39	300	300	300	3	10	0.3
40	300	300	300	10	10	0.05
41	300	300	300	10	10	0.1
42	300	300	300	10	10	0.3
43	300	300	300	3	50	0.05
44	300	300	300	3	50	0.1
45	300	300	300	3	50	0.3
46	300	300	300	10	50	0.05
47	300	300	300	10	50	0.1
48	300	300	300	10	50	0.3

Table 5: Parameters Settings and their IDs.

- rep_MC has two levels: 10 and 50 repetitions.

Thus, there are 48 parameters settings for each environment. They are listed and identified in Table 5.

The analyses performed have two phases. First, an analysis of variance (ANOVA) was performed in order to determine the sensitivity of DTC to its parameters. For that sake, the parameters settings provided in Table 5 were adopted. The hypotheses considered were the following:

Null Hypothesis : there is no significant difference among the averages of the offline errors (*oe*). If there are differences, they are due to random effects.

Alternative Hypothesis : there is a combination of factor values for which the averages of the offline errors (*oe*) are significatively different and they are not due to random effects.

As the results (offline errors) do not follow a normal distribution, the Kruskal-Wallis test was applied, to perform the ANOVA and then the Turkey method in order to determine the experimental conditions for which significant differences exist. The results obtained by the ANOVA proved the Null Hypothesis for several combinations of parameters. However, the Alternative Hypothesis was proved, too. Figures 10, 11, 12, 13, 14 and 15 show in the x-axis the statistics used by the Turkey method to determine significant differences and the y-axis indicates the ID for the combination of parameter values (corresponding to Table 5). When the two intervals are not overlapping, the results obtained with these combinations of parameter values are significantly different; otherwise, they are not significantly different. For instance, Figure 10 shows the ANOVA for STCG$_{12nc}$. Here, it can be seen that the results obtained with the parameters setting 01 has no significant differences with respect to the results obtained with the parameters settings 02 to 14 (the interval for 01 is overlapped with the intervals from 02 to 14).

In the second phase of the statistical analysis, the box plot method was adopted to visualize the distribution of the offline errors for each environment. This allowed to determine the robustness of DTC. Figures 16, 17 and 18 show in the x-axis the ID for a combination of parameter values (corresponding to Table 5) and the y-axis indicates the offline errors for each environment.

Thus, these two phases are meant to answer the following questions:

1. Is prob$_{diff-CD4}$ responsible for causing the significant differences in the results?

2. Is the size of the effector populations responsible for causing the significant differences in the results?

3. Is the number of memory cells responsible for causing the significant differences in the results?

4. Is the number of activation of MC (rep_MC) responsible for causing the significant differences in the results?

After the statistical analysis of the results obtained by DTC, for the six environments, with respect to the parameters settings provided in Table 5, the following general conclusions can be derived:

- When the size of the populations and the number of activations of MC are fixed, to vary the probability of application of the differentiation operator for CD4 does not produce significant differences on the results. Thus, $\text{prob}_{diff-CD4}$ is not considered as a key parameter on the performance of the DTC algorithm. However, although this is not a critical parameter, it can be seen from Figures 16, 17 and 18 that better medians for STCG_{12nc} (when the sizes of CD4 and CD8 are 50, 100 or 200), STCG_{10c}, STCG_{20c}, STCG_{20nc} and MPB_{50} (with 3 memory cells) were obtained when this probability is high.

- When the sizes of both CD4 and CD8 grow, the results show significant differences. Thus, these parameters have a high impact on the performance of the proposed DTC approach.

- The size of MC does not produce significant differences, when the rest of the parameters remain fixed. When the size of MC is lower, rep_MC does not seem to affect or improve the results as the sizes of MC and rep_MC are larger. This means that DTC does not stagnate.

- To vary the number of activations of MC, when the rest of the parameters remain fixed, does not produce significant differences, except for MPB_{50} when using a size of 300 cells for both CD4 and CD8.

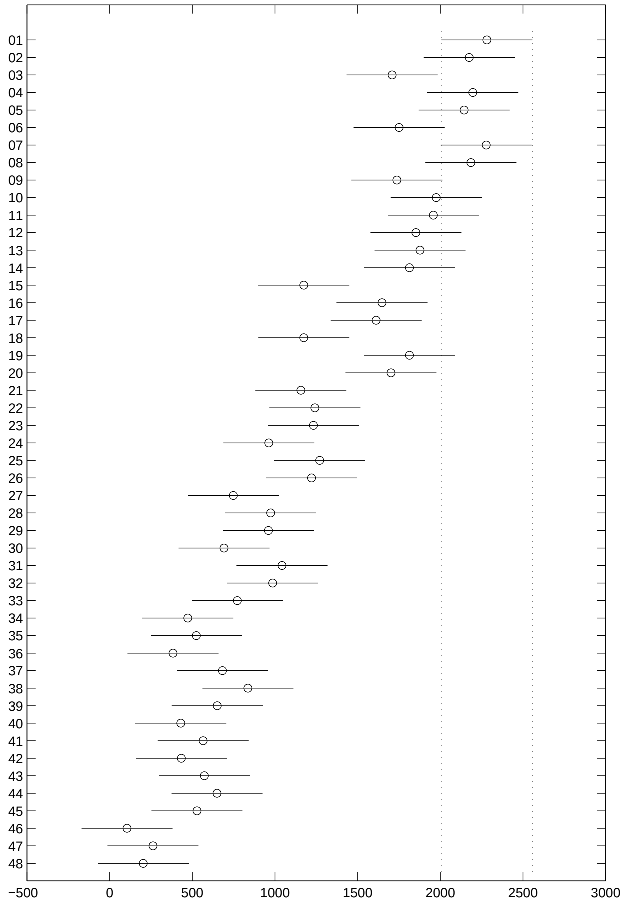


Figure 10: ANOVA for STCG_{12nc}.

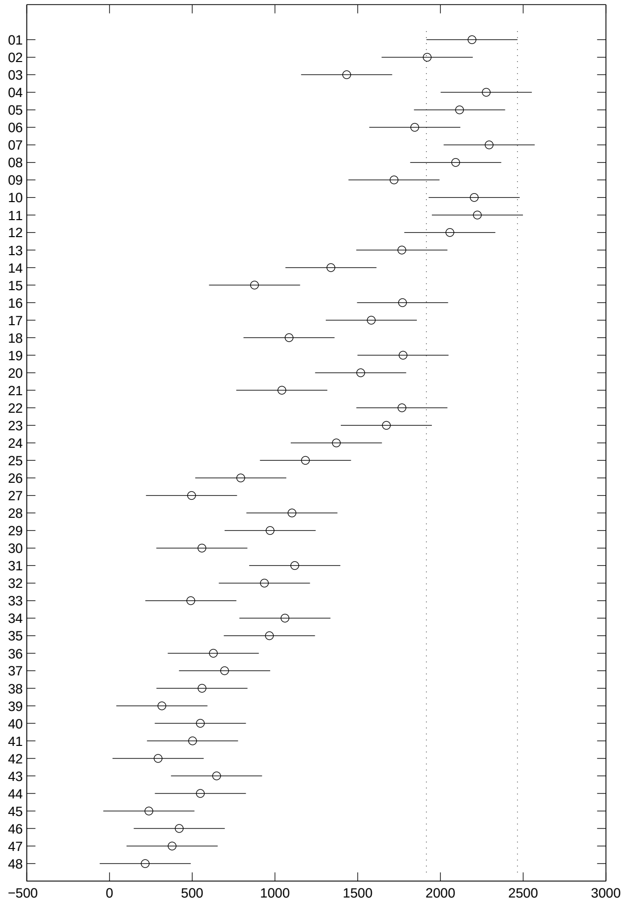


Figure 11: ANOVA for STCG_{20nc}.

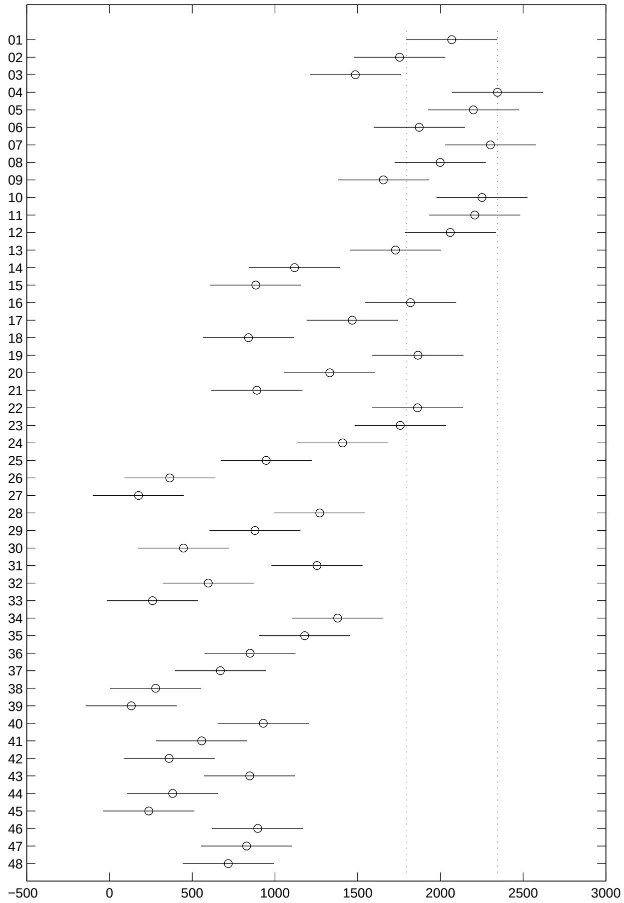


Figure 12: ANOVA for STCG_{10c}.

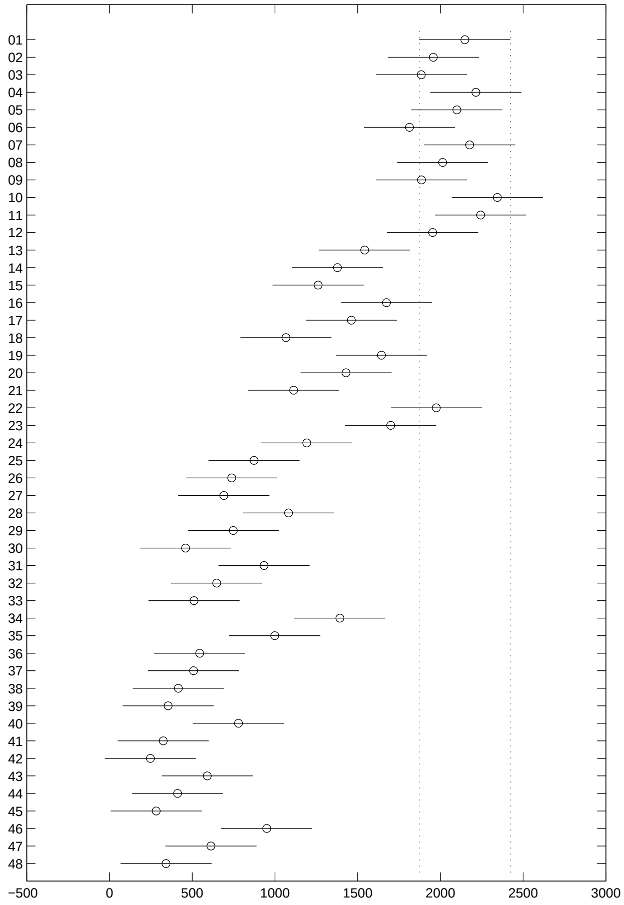


Figure 13: ANOVA for STCG_{20c}.

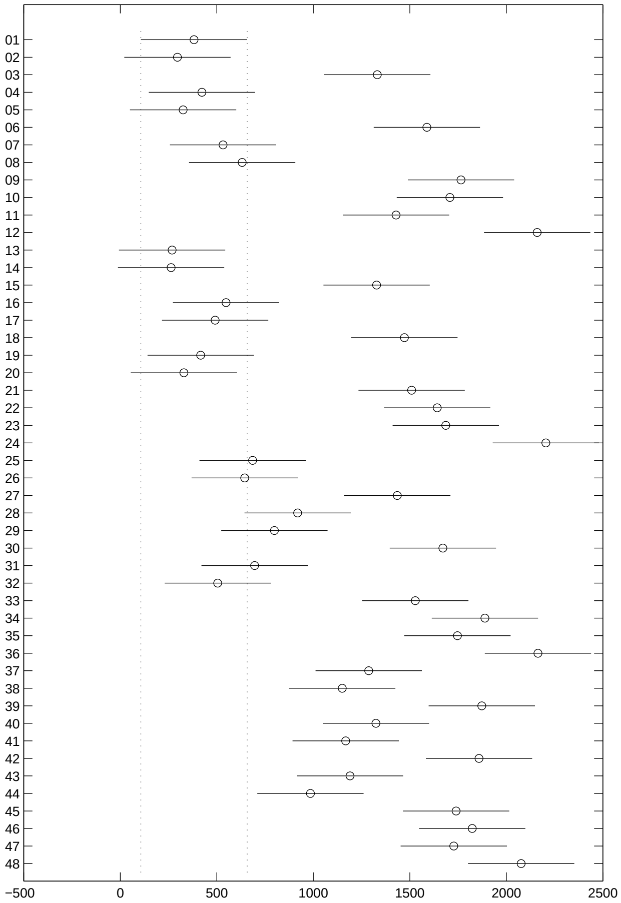


Figure 14: ANOVA for MPB_5.

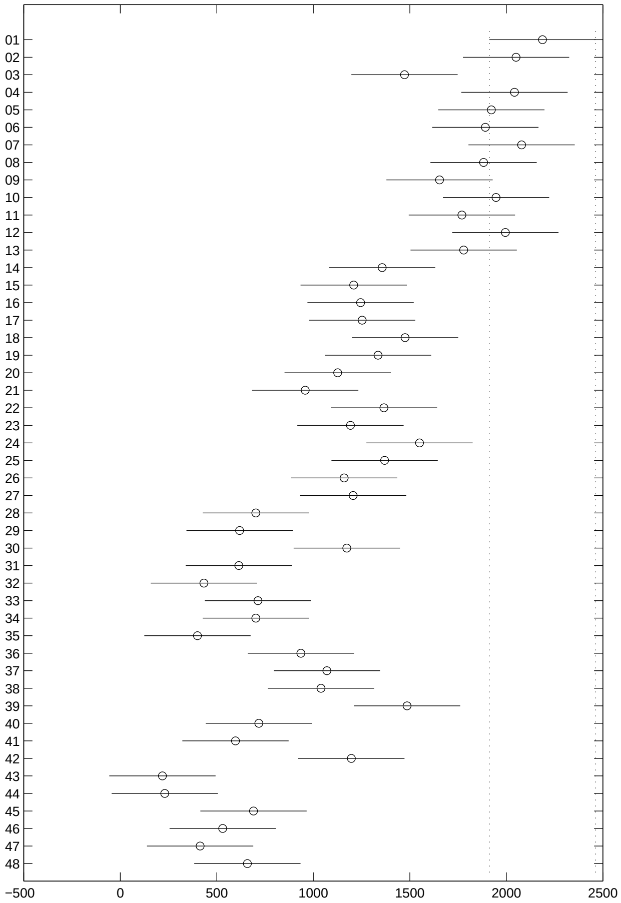
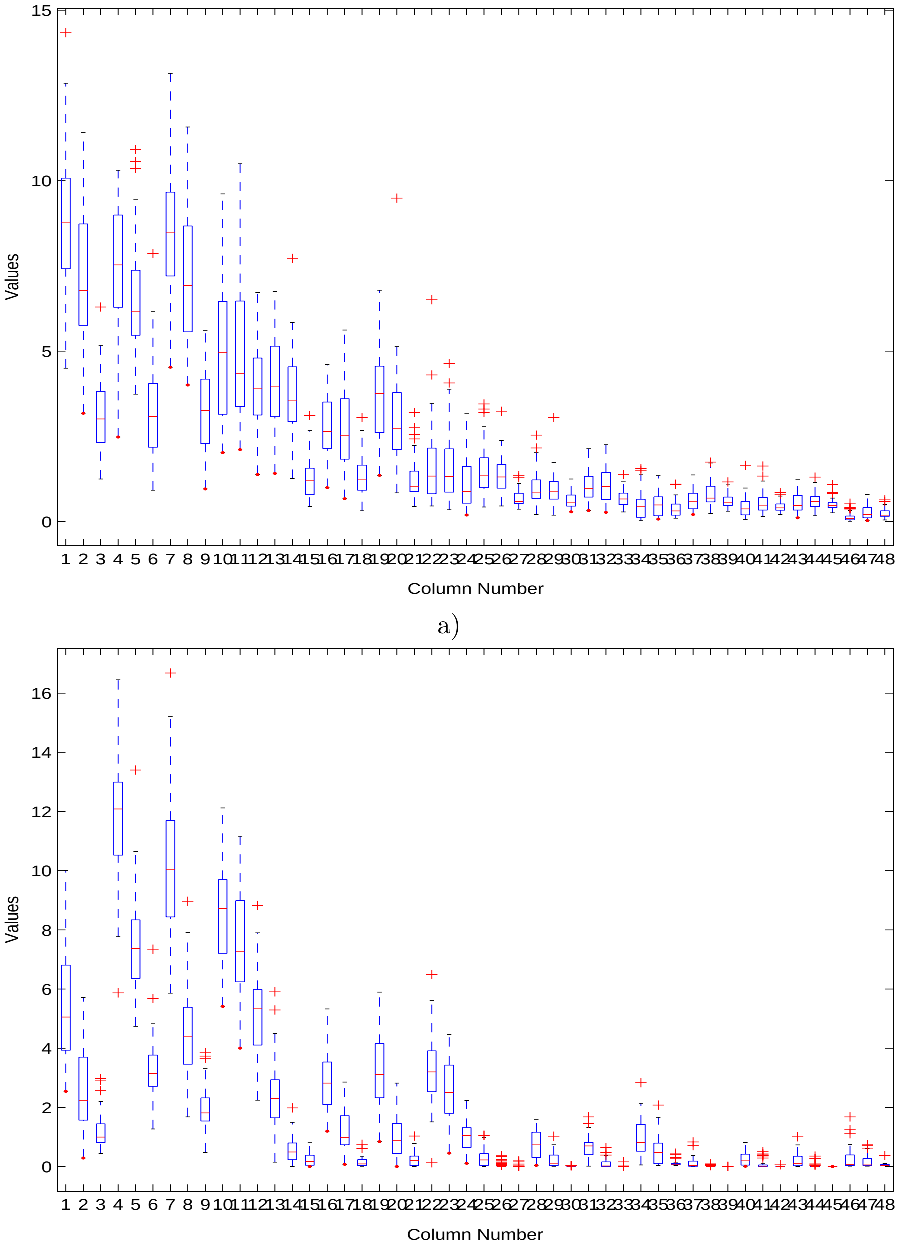


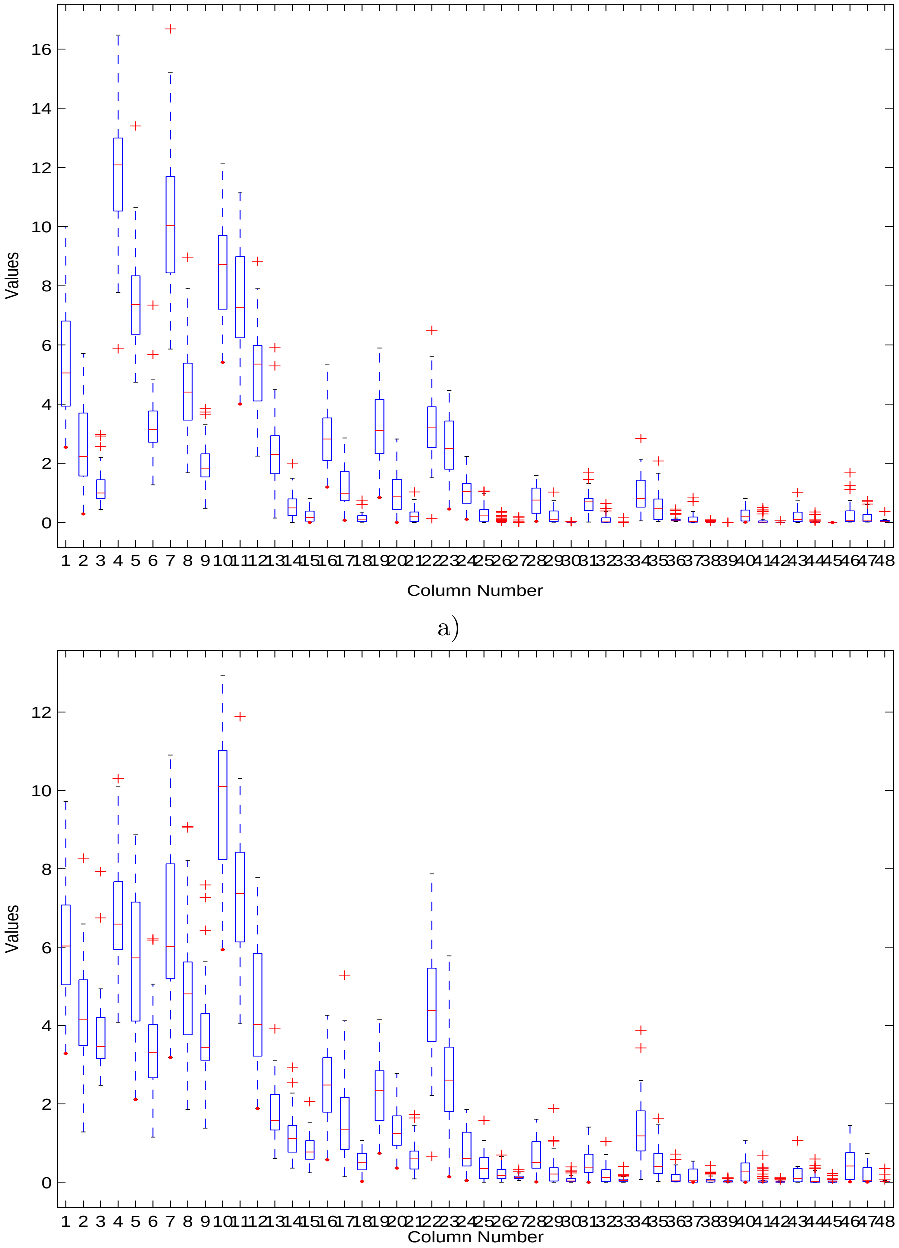
Figure 15: ANOVA for MPB_{50}.



a)

b)

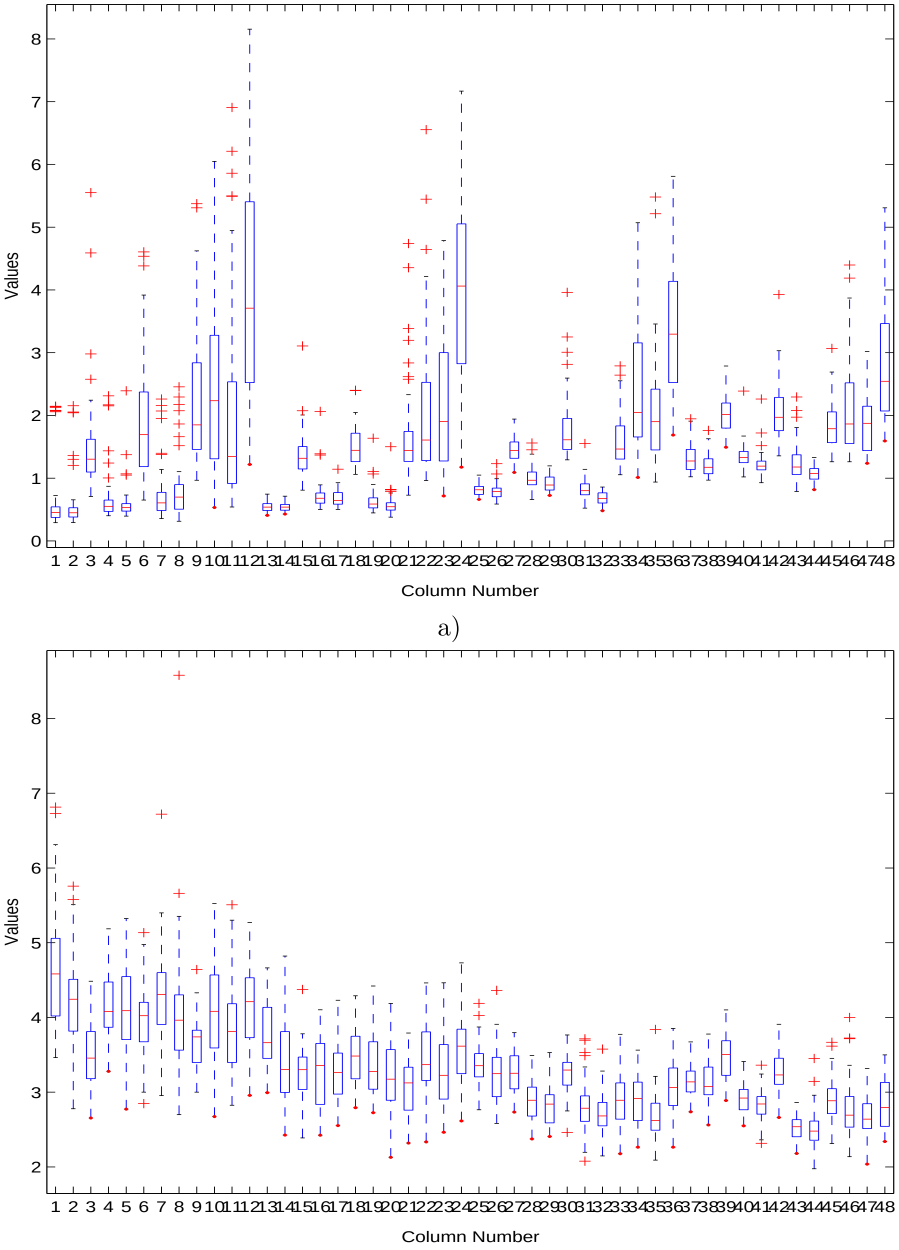
Figure 16: Boxplot: a) STCG_{12nc}, b) STCG_{10c}.



a)

b)

Figure 17: Boxplot: a) STCG$_{10c}$, b) STCG$_{20c}$.



a)

b)

Figure 18: Boxplot: a) MPB_5, b) MPB_{50}.

Thus, the parameters can be ranked with respect to their influence on the performance of DTC. In the first place, it is considered the size of both CD4 and CD8, since they have the highest influence on the performance of DTC. In second place, it is considered $\text{prob}_{diff-CD4}$. Finally, it is considered the number of memory cells and the number of activations for MC. For five of the six environments studied (all but MPB_5) the best results were found when using 200 and 300 effector cells. On the other hand, for MPB_5, DTC presented a better performance with 50 and 100 effector cells. For all the environments, except for MPB_{50}, DTC is robust when considering the appropriate parameters settings.

These facts make us to recommend, in general, the following values for the parameters required by the DTC algorithm:

1 For environments in which the differences among heights are big, or when the landscape has many local optima

- $|\text{VC}|$= 300 cells
- $|\text{CD4}|$= 200 to 300 cells
- $|\text{CD8}|$= 200 to 300 cells
- $|\text{MC}|$= 3 or 10 cells
- rep_MC = 10 or 50
- $\text{prob}_{diff-CD4} = 0.3$

2 For environments in which the differences among heights and the optimum are small or when the landscape has only a few local optima

- $|\text{VC}|$= 300 cells
- $|\text{CD4}|$= 50 to 100 cells
- $|\text{CD8}|$= 50 to 100 cells
- $|\text{MC}|$= 3 cells
- rep_MC = 10
- $\text{prob}_{diff-CD4} = 0.05$ or 0.1

9 Conclusions and Future Work

In this paper, it has been presented an adaptation of an existing artificial immune system based on the T-Cell model, to the solution of dynamic optimization problems. The model that has been adopted is inspired on the processes experienced by the T-Cells within our immune system.

The proposed approach has been tested with six environments taken from a benchmark of dynamic functions commonly adopted in the specialized literature. The results obtained by DTC were very competitive, even when it is not a multipopulation algorithm, resulting better in five of the six cases adopted, than those generated by other algorithms from the state-of-the-art with respect to which it was compared.

Based on this evidence, the authors believe that the proposed approach (called DTC) can be a viable alternative to solve dynamic optimization problems. Additionally, DTC is able to maintain diversity without using any extra mechanisms.

As part of the future work to be undertaken, it would be interesting to study mechanisms that allow to reduce the number of parameters that DTC requires. It would also be interesting to explore techniques that can increase the robustness and effectivity of DTC. For that sake, it would be important to study, for example, the effect of the number of clones in the search process.

Acknowledgements

The authors would like to thank Dr. Krzysztof Trojanowski (Institute of Computer Science, Polish Academy of Sciences), for kindly providing the source code of the Simple Test-Case Generator (STCG). The authors also thank the anonymous reviewers and the Editor-in-Chief for their valuable comments which greatly helped them to improve the contents of this paper.

The third author acknowledges support from CONACyT project number 103570.

References

[1] Victoria S. Aragón, Susana C. Esquivel, and Carlos A. Coello Coello. Artificial Immune System for Solving Global Optimization Problems. In *XIV Congreso Argentino en Ciencias de la Computacion (CACIC 2008)*, pages 637–647, 2008. Chilecito, La Rioja, Argentina.

[2] Michal Bereta and Tadeusz Burczynski. Immune K-means and negative selection algorithms for data analysis. *Information Sciences*, 179(10):1407–1425, April 29 2009.

[3] T. Blackwell and J. Branke. Multiswarms, exclusion, and anti-convergence in dynamic environments. *IEEE Transactions on Evolutionary Computation*, 10(4):459–472, 2006.

[4] Jürgen Branke. Memory Enhanced Evolutionary Algorithms for Changing Optimization Problems. In *Proceedings of the 1999 Congress on Evolutionary Computation (CEC'99)*, volume 3, pages 1875–1882, Washington, D.C., July 1999. IEEE Service Center.

[5] Jürgen Branke. *Evolutionary Optimization in Dynamic Environments.* Kluwer Academic Publishers, 2002.

[6] Peter Bretscher and Melvin Cohn. A theory of self-nonself discrimination. *Science*, 169(3950):1042–1049, September 1970.

[7] Dipankar Dasgupta and Luis Fernando Ni no. *Immunological Computation: Theory and Applications*. Auerbach Publications, Boston, Massachusetts, USA, 2008.

[8] Weilin Du and Bin Li. Multi-strategy ensemble particle swarm optimization for dynamic optimization. *Information Sciences*, 178(15):3096–3109, August 1 2008.

[9] Alessio Gaspar and Philippe Collard. From GAs to Artificial Immune Systems: Improving Adaptation in Time Dependent Optimization. In *Proceedings of the 1999 IEEE Congress on Evolutionary Computation (CEC'99)*, volume 3, pages 1859–1866. IEEE Service Center, July 1999.

[10] Maoguo Gong, Licheng Jiao, and Lining Zhang. Baldwinian learning in clonal selection algorithm for optimization. *Information Sciences*, 180(8):1218–1236, April 15 2010.

[11] Johnny Kelsey and Jon Timmis. Immune Inspired Somatic Contiguous Hypermutation for Function Optimisation. In Erick Cantú-Paz et al., editor, *Genetic and Evolutionary Computation - GECCO 2003, Genetic and Evolutionary Computation Conference*, pages 207–218, Chicago, Illinois, USA, July 2003. Springer. Lecture Notes in Computer Science Vol. 2723.

[12] James Kennedy and Russell C. Eberhart. *Swarm Intelligence*. Morgan Kaufmann Publishers, San Francisco, California, 2001.

[13] Xiaodong Li, Jürgen Branke, and Tim Blackwell. Particle swarm with speciation and adaptation in a dynamic environment. In Maarten Keijzer et al., editor, *2006 Genetic and Evolutionary Computation Conference (GECCO'2006)*, volume 1, pages 51–58, Seattle, Washington, USA, July 2006. ACM Press. ISBN 1-59593-186-4.

[14] David Male, Jonathan Brostoff, David B. Roth, and Ivan Roitt. *Inmunology*. Mosby, seventh edition, 2006.

[15] P. Matzinger. Tolerance, danger and the extended family. *Annual Review of Immunology*, 12:991–1045, April 1994.

[16] Zbigniew Michalewicz and David B. Fogel. *How to Solve It: Modern Heuristics*. Springer, second edition, 2004.

[17] Leandro N. de Castro and Jon Timmis. An artificial immune network for multimodal function optimization. In *Proccedings of the 2002 IEEE Congress on Evolutionary Computation (CEC'2002)*, volume 1, pages 669–674, Honolulu, Hawaii, USA, May 2002. IEEE Service Center.

[18] Nikolaos Nanas and Anne De Roeck. Multimodal Dynamic Optimization: From Evolutionary Algorithms to Artificial Immune Systems. In Leandro Nunes de Castro, Fernando José Von Zuben, and Helder Knidel, editors, *Artificial Immune Systems, 6th International Conference, ICARIS*

2007, pages 13–24. Springer. Lecture Notes in Computer Science Vol. 4628, Santos, Brazil, August 2007.

[19] Leandro Nunes de Castro and Jonathan Timmis. *Artificial Immune Systems: A New Computational Intelligence Approach*. Springer-Verlag, New York, 2002.

[20] Leandro Nunes de Castro and Fernando J. Von Zuben. The Clonal Selection Algorithm with Engineering Applications. In *Proceedings of Genetic and Evolutionary Computation Conference, Workshop on Artificial Immune Systems and Their Applications*, pages 36–37, July 2000.

[21] Fabricio Olivetti de França, Fernando J. Von Zuben, and Leandro Nunes de Castro. An artificial immune network for multimodal function optimization on dynamic environments. In *Proceedings of the 2005 Genetic and Evolutionary Computation Conference (GECCO'2005)*, pages 289–296, New York, NY, USA, 2005. ACM Press.

[22] Benjamin A. Schwarz and Avinash Bhandoola. Trafficking from the bone marrow to the thymus: a prerequisite for thymopoiesis. *Immunological Reviews*, 209(1):47–57, 2006.

[23] Krzysztof Trojanowski. *Evolutionary Algorithms with Redundant Genetic Material for Non - Stationary Environments*. PhD thesis, Institute of Computer Science, Warsaw University of Technology, Poland, 1994.

[24] Krzysztof Trojanowski. Clonal Selection Principle Based Approach to Non-Stationary Optimization Tasks. In Jarosław Arabas, editor, *Evolutionary Computation and Global Optimization 2006*, number 156 in Prace Naukowe, Elektronika, z.156, pages 375–384. Warsaw University of Technology Publishing House, 2006.

[25] Krzysztof Trojanowski. B-Cell Algorithm as a Parallel Approach to Optimization of Moving Peaks Benchmark Tasks. In *International Conference on Computer Information Systems and Industrial Management Applications*, pages 143–148, Los Alamitos, CA, USA, 2007. IEEE Computer Society.

[26] Krzysztof Trojanowski. Clonal Selection Approach with Mutations Based on Symmetric alpha Stable Distributions for Non-stationary Optimization Tasks. In *ICANNGA '07: Proceedings of the 8th international conference on Adaptive and Natural Computing Algorithms, Part I*, pages 184–193, Berlin, Heidelberg, 2007. Springer-Verlag.

[27] Krzysztof Trojanowski. Properties of Quantum Particles in Multi-Swarms for Dynamic Optimization. *Fundamenta Informaticae*, 95(2-3):349–380, 2009.

[28] Krzysztof Trojanowski and Zbigniew Michalewicz. Searching for Optima in Non-Stationary Environments. In Peter J. Angeline, Zbigniew Michalewicz, Marc Schoenauer, Xin Yao, and Ali Zalzala, editors, *Proceedings of the 1999 IEEE Congress on Evolutionary Computation (CEC'99)*, volume 3, pages 1843–1850, Washington, USA, July 7-9, 1999. IEEE Press, Piscataway, NJ, USA.

[29] Krzysztof Trojanowski and Slawomir T. Wierzchon. Studying Properties of Multipopulation Heuristic Approach to Non-Stationary Optimisation Tasks. In Mieczyslaw A. Klopotek, Slawomir T. Wierzchon, and Krzysztof Trojanowski, editors, *Intelligent Information Processing and Web Mining, Proceedings of the International IIS: IIPWM'03*, Advances in Soft Computing, pages 23–32. Springer, 2003.

[30] Krzysztof Trojanowski and Slawomir T. Wierzchoń. Immune-based algorithms for dynamic optimization. *Information Sciences*, 179(10):1495–1515, April 29 2009.

[31] Huan Xu, Jianjun Wang, and Hyoung Joong Kim. Near-optimal solution to pair-wise LSB matching via an immune programming strategy. *Information Sciences*, 180(8):1201–1217, April 15 2010.

[32] Xinquan Zuo, Hongwei Mo, and Jianping Wu. A robust scheduling method based on a multi-objective immune algorithm. *Information Sciences*, 179(19):3359–3369, September 9 2009.