A Modified Version of a T-Cell Algorithm for Constrained 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

March 2, 2010

Abstract

In this paper, we present a heuristic inspired on the T-Cell model of the immune system (i.e., an artificial immune system). The proposed approach (called T-Cell) is used for solving constrained (numerical) optimization problems, and is validated using several test functions taken from the specialized literature on evolutionary optimization. Additionally, several engineering optimization problems are also used for assessing the performance of the proposed approach. Results are compared with respect to approaches representative of the state-of-the-art in constrained evolutionary optimization.

Keywords: Artificial immune systems, constrained optimization, metaheuristics, engineering optimization.

*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 3175 CNRS at CINVESTAV-IPN.

1 Introduction

In recent years, a bio-inspired metaheuristic known as the "artificial immune system" (AIS) has gained popularity in a wide variety of tasks [34]. AISs are inspired on our natural immune system, which has a number of very interesting features, from a computational point of view, that make it a very good candidate to be modelled in a computer. For example, it is a distributed system, it is fault-tolerant, it has memory, it is able to distinguish between its own components and those which are foreign, and it learns by experience.

AISs have been used for solving optimization problems (see for example [13, 17, 42]), but in most cases, it has been applied only to unconstrained problems. The main reason for this is that this approach, in its original form (like in the case of evolutionary algorithms) is an unconstrained search and optimization technique. Relatively few proposals exist for coupling constraint-handling mechanisms to an AIS (see for example [14]), in spite of the importance that constraints have in real-world problems. In this paper, we precisely focus on solving constrained optimization problems with an AIS that we have proposed, and which we believe that can be a viable alternative for solving engineering optimization problems.

The remainder of the paper is organized as follows. In Section 2, we define the general type of problem we want to solve. Sections 3 and 4 provide some general concepts regarding the existing AISs as well as the previous work related to our own, respectively. In Section 5, we describe our proposed AIS, which is based on the T-Cell Model. In Section 6, we present our experimental setup, our results and we discuss them. Finally, in Section 7, we present our conclusions and some possible paths for future research.

2 Statement of the Problem

We are interested in solving problems of the form[1]:

minimize

$$f(X) \tag{1}$$

subject to

$$g_j(X) \leq 0 \qquad j = 1, \ldots, m \tag{2}$$

$$h_k(X) = 0 \qquad k = 1, \ldots, p \tag{3}$$

$$x_i^l \leq x_i \leq x_i^u \qquad i = 1, \ldots, n \tag{4}$$

here f designates the objective function and $X = (x_1, x_2, \ldots, x_n)^T$ is a vector containing the design variables. The remaining functions correspond to inequality constraints (g), equality constraints (h) and side constraints with lower and

[1] Without loss of generality, we will assume only minimization problems.

upper limits indicated by the superscripts l and u, respectively. Both the objective function and the constraints could be linear or nonlinear.

When an inequality constraint takes a zero value at the optimum, we say that it is **active**. By definition, all the equality constraints are active.

3 AIS Models

According to [18] the main computational models of an Artificial Immune System in current use are: Negative Selection, Clonal Selection and Immune Network Models. Each of them is briefly described next.

Forrest et al. [16] proposed the Negative Selection model for detection of changes in an environment. This model is based on the discrimination principle that the immune system adopts to distinguish between self and nonself. This model generates random detectors and discards the detectors that are unable to recognize themselves. Thus, it maintains the detectors that identify any nonself. It performs a probabilistic detection and it is robust because it searches any foreign action instead of a particular action. Typical applications of negative selection include computer security [16, 20] and classification [19], among others.

The Immune Network Model was proposed by Jerne [26], and it is a mathematical model of the immune system. This model assumes that lymphocytes are an interconnected network, and the dynamics of such lymphocytes is simulated using differential equations. Several computational models have been derived from this mathematical theory (see for example [21, 25]). Typical applications of immune network models are: detection of gene promoter sequences [22], data mining [23], diagnosis [24] and cluster analysis [39], among others.

Clonal Selection is based on the way in which both B-cells and T-cells adapt in order to match and kill foreign cells [34]. Clonal Selection involves: 1) the AIS' ability to adapt its B-cells to new types of antigens and 2) the affinity maturation by hypermutation. CLONALG is a popular AIS based on clonal selection proposed by Nunes de Castro and Von Zuben [32], which was originally used to solve pattern recognition and multimodal optimization problems. A few extensions of CLONALG are available for dealing with constrained optimization problems. CLONALG works in the following way: first, it creates a random population of antibodies, it sorts such antibodies according to some affinity function (this affinity function is analogous to the fitness function used in evolutionary algorithms), it clones them, it hypermutates each clone, and it selects the antibodies with greatest affinity. Then, it replaces the worst antibodies in the population by antibodies that are randomly generated. Typical applications of clonal selection are described in [32, 33, 40].

4 Previous Related Work

As indicated before, the use of AIS for solving constrained (numerical) optimization problems is less commom in the literature, when compared to other

nature-inspired techniques. The most relevant references in this regard are briefly discussed next.

Hajela and Yoo [41, 42] proposed a hybrid between a Genetic Algorithm (GA) and an AIS for solving constrained optimization problems. This approach works on two populations. The first is composed by the antigens (which are the feasible solutions to the problem), and the other is composed by the antibodies (which are the infeasible solutions to the problem). The idea is to have a GA embedded into another GA. The outer GA performs the optimization of the original (constrained) problem. The second GA uses as its fitness function a Hamming distance so that the antibodies are evolved to become very similar to the antigens, without becoming identical. An interesting aspect of this work was that the infeasible individuals would normally become feasible as a consequence of the evolutionary process performed which, however, did not use the amount of constraint violation to guide the search. This approach was successfully applied to some structural optimization problems.

Coello Coello and Cruz-Cortés [13] proposed an extension of Hajela and Yoo's algorithm. In this proposal, no penalty function is needed, and some extra mechanisms are defined to allow the approach to work in cases in which there are no feasible solutions in the initial population. Additionally, the authors proposed a parallel version of the algorithm and validated it using some standard test functions reported in the specialized literature.

Balicki [4] made a proposal very similar to the approach of Coello Coello and Cruz-Cortés. Its main difference is the way in which the antibodies' fitness is computed. In this case, a ranking procedure is adopted. This approach was validated using a constrained three-objective optimization problem.

Luh and Chueh [17, 29] proposed an algorithm (called CMOIA, or Constrained Multi Objective Immune Algorithm) for solving constrained multiobjective optimization problems. In this case, the antibodies are the potential solutions to the problem, whereas antigens are the objective functions. CMOIA transforms a constrained problem into an unconstrained one by associating an interleukine (IL) value with all the constraints violated. IL is a function of both the number of constraints violated and the total magnitude of this constraint violation. Then, feasible individuals are rewarded and infeasible individuals are penalized. Other features of the approach are based on clonal selection theory and other immunological mechanisms. CMOIA was evaluated using six test functions and two structural optimization problems.

Coello Coello and Cruz-Cortés [14] proposed an algorithm based on clonal selection theory for solving constrained optimization problems. The authors experimented with both binary and real-numbers encoding, considering Gaussian-distributed and Cauchy-distributed mutations. Furthermore, they proposed a controlled and uniform mutation operator. This approach was tested with a set of 13 test functions taken from the specialized literature on evolutionary constrained optimization.

Bernardino et al. [9, 10] proposed a genetic algorithm hybridized with an artificial immune system (AIS-GA). The AIS is inspired on the clonal selection principle and it is embedded into a standard GA search engine in order to

help moving the population into the feasible region. AIS-GA uses binary tournaments in which the rules of the tournament are those normally adopted for constrained problems. The authors argued that their AIS-GA performed very well in problems presenting continuous design variables, that it reached good results in problems with mixed design variables and that it presented a poor performance in problems with discrete design variables. The authors also presented an AIS-GA with a clearing procedure, which is called AIS-GA^C. This procedure is applied to the union of the new population and the previous one, in order to create the new population. Both approaches are applied to six mechanical engineering optimization problems.

Bernardino et al. presented in [8] a modified version of the algorithms in [9] and [10]. The main difference with respect to these previous versions is that the new one calculates the affinity taking into account the sum of the constraint violations.

Previous versions of the T-Cell Algorithm presented in this paper have been published before (see [1, 2, 3]) and validated with constrained optimization problems. There are, however, significant differences between this paper and those previous publications, which we explain next.

The algorithm proposed in [1] uses a similar mutation operator for the memory cells, but not the same as here; additionally, it also keeps fixed the number of cells in the virgin cells during all the search process. The algorithms proposed in [2, 3] utilize random probabilities for the mutation operators of the effector cells. Also, those previous approaches keep fixed the number of cells in the virgin cells during all the search process. It is also worth noting that these previous approaches had not been applied to engineering optimization problems such as those reported here. Finally, another important difference with respect to these previous publications is that here, we do not adopt any dynamic tolerance factor for the constraints. Such factor allows to expand the feasible region such that equality constraints can be handled in a better way, but has the disadvantage of adding one extra parameter to our approach [1, 2, 3].

5 Our Proposed Approach

In this paper we propose an adaptive immune system model, called TCELL, which is based on the immune responses mediated by the T-cell. Our model considers many of the processes that suffer the T-cells from their origin in the hematopoietic stem cells in the bone marrow until they become memory cells.

The T-cells belong to a group of white blood cells known as lymphocytes. They play a central role in cell-mediated immunity. They present a special receptor on their cell surface called T-cell receptors (TCR[2]). All T-cells originate from hematopoietic stem cells in the bone marrow. Hematopoietic progenitor derived from hematopoietic stem cells populate the thymus and expand by cell division to generate a large population of immature thymocyte [37].

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

T-cells can be divided into three groups, according to their maturation or development level: virgin cells, effector cells and memory cells (phylogenies of the T cells [15]). Virgin cells are those that were never activated (this means that they did not suffer proliferation or differentiation).

Effector cells are a type of cells that can be activated by the recognition of an antigen [11, 30]. The activation of an effector cell implies that it will be proliferated and differentiated. The proliferation process has as its goal to replicate the cells and the differentiation process changes the clones in order to acquire specialized functional properties.

Finally, the memory cells are cells that persist into the host even when the infection or danger has been overtaken so that, in the future, they are able to be stimulated by the same (or a similar) antigen. Usually, they respond through proliferation and differentiation, faster with a low dosage of an antigen than the memory B cells. Note that although effector and memory cells are proliferated, they do not suffer somatic hypermutation.

Thus, during the immunological response, the T-cells pass through different phases: initiation, reaction and elimination. After the initiation phase, the virgin cells become effector cells. They are activated (the cells change in order to improve) and undergo a process called *apoptosis*. This process eliminates any undesirable cells. The surviving cells become memory cells.

The previous concepts served as our inspiration to develop a T-Cell Algorithm (when considering constrained problems). Our approach operates on four populations: (1) Virgin Cells (VC), (2) Effector Cells (EC), which are subdivided into: (2.a) Effector Feasible Cells (EC_f) and (2.b) Effector Infeasible Cells (EC_inf), and (3) Memory Cells (MC). Each of them has a specific function. VC has as its main goal to provide diversity. EC tries to explore the conflicting zones of the search space. MC has to explore the neighborhood of the best solutions found so far. The *apoptosis* is modeled through the insertion of VC into EC and EC into MC. VC and EC represent their cells with binary strings using Gray coding and MC adopt vectors of real numbers. These representations were chosen in order to be benefited with their corresponding properties. This also has a biological justification, since in biology, memory cells have a heterogeneous structure such as the one adopted in our model.

5.1 Handling Constraints

In our T-Cell Algorithm, the constraint-handling method needs to calculate, for each cell (solution), regardless of the population to which it belongs, the following: 1) the value of each constraint function, 2) the sum of constraint violations (sum_res)[3] and 3) the value of the objective function (only if the cell is feasible).

[3]This is a positive value determined by $g_i(x)^+$ for $i = 1, \ldots, m$ and $|h_k(x)|$ for $k = 1, \ldots, p$.

5.2 Incorporating Domain Knowledge

As indicated before, the effector cells are subdivided into EC_f and EC_inf (feasible and infeasible solutions, respectively) in order to explore the boundary between the feasible and the infeasible regions. Also, we introduce domain knowledge through the mutation operators, which modify the decision variables involved in a particular constraint (either the constraint with the highest violation, or the one with the most negative value, depending on whether the cell is infeasible or not, respectively).

5.3 Mutation Operators

Each population that reacts (EC_f, EC_inf and MC) has its own mutation operator. These operators are described next.

The mutation operator for EC_inf works in the following way: first, it identifies the most highly violated constraint, say c. If this constraint value (c) is larger than sum_res divided by the total number of constraints, then we change each bit from each decision variable involved in c with probability $prob_{mut}$. Otherwise, we change each bit from one decision variable involved in c, randomly selected, with probability $prob_{mut}$.

There are two mutation operators for EC_f, which generate two mutated cells (one per operator). The best of these cells passes to the following iteration. These operators work in the following way:

First operator: it identifies the constraint with the most negative value (keep in mind that this population has only feasible cells), and changes each bit (from 0 to 1 or viceversa) from each decision variable involved in that constraint, with probability $prob_{mut}$. This operator tries to reduce the distance between the cell and the boundary with the infeasible region. In other words, the operator tries to find solutions close to the frontier between the feasible and the infeasible regions.

Second operator: it changes each bit from all the decision variables, with probability $prob_{mut}$. This operator tries to perform a global search.

If, after applying either of the mutation operators, a cell becomes feasible, it is inserted in EC_f according to an elitist selection. Otherwise, if after applying the mutation operator, a cell becomes infeasible, it is inserted into EC_inf according to an elitist selection (see Subsection 5.4).

The mutation operator for MC applies the following equation to all decision variables:

$$x^{'} = x \pm \left(\frac{U(0,1)(l_u - l_l)}{10 iter |const||dv|} \right)^{U(0,2)} \tag{5}$$

where x and $x^{'}$ are the original and mutated decision variables, respectively. $U(0,1)$ and $U(0,2)$ refer to a randomly generated number, produced with a uniform distribution in the range [0,1] and [0,2], respectively. l_u and l_l are the upper and lower limits of x. $|const|$ refers to the number of constraints of the

problem. $|dv|$ refers to the number of decision variables of the problem and $iter$ is the current outer iteration number.

These operators work on continuous decision variables, but sometimes, the problems include discrete variables. In those cases, when the decision variable is discrete, we used the following equation before evaluating the constraints and/or the objective function:

$$dv' = \begin{cases} l_j & if|l_j - dv| \leq |l_{j+1} - dv| \\ l_{j+1} & otherwise \end{cases} \tag{6}$$

where dv is the continuous variable, dv' is the discrete variable corresponding to dv, $l_k \in S$ and S is the sorted set of discrete values for dv'.

5.4 Replacement Mechanisms

The replacement mechanisms are always applied in an elitist way, both within a population and between different populations. They take into account the value of the objective function or the sum of constraints violation, depending on whether the cell is feasible or infeasible, respectively. Additionally, it always considers a feasible cell better than an infeasible one.

When a cell from EC_f or EC_inf has to be inserted into MC, the cell first has to be converted into a real-value vector through the application of the following formula:

$$dv_j = l_{lj} + \frac{\sum_{i=0}^{L_j} 2^{L_j - i} dv'_{ij}(l_{uj} - l_{lj})}{2_j^L - 1} \tag{7}$$

where dv_j is the j^{th} decision variable with $j = 1, \ldots, \mid dv \mid$ ($\mid dv \mid$ is the number of decision variables), L_j is the number of bits for the j^{th} decision variable, l_{uj} and l_{lj} are the upper and lower bounds for the continuous decision variable dv_j, and dv'_{ij} is the i^{th}-bit of the binary string that represents dv_j.

The algorithm works in the following way. At the beginning, virgin cells are initialized in a random way. Then, the constraints and the objective function are evaluated, in order to determine which cells are feasible and which are infeasible. Next, they are divided into the two populations EC_f (feasible solutions) and EC_inf (infeasible solutions) The size of each population is fixed. But, at first, EC_f and EC_inf are empty. If, at the beginning, EC_f can not be filled up with feasible cells from VC, the size of EC_f must be less than a maximum (predefined) value. In this case, further iterations could fill up EC_f. This situation occurs for EC_inf, too, but considering infeasible cells from VC. Once EC_f and EC_inf have cells, they undergo mutation (using the operators previously described).

The algorithm tries first to find feasible solutions using infeasible cells from EC_inf. Then, all the feasible cells generated by the application of the mutation operator for EC_inf do the following: 1) replace the cells with highest objective function values in EC_f, if they are better or 2) complete EC_f, if the size of EC_f

is less than its maximum (predefined) value. Then, the T-Cell tries to find solutions close to the boundary between the feasible and infeasible regions by using feasible cells from EC_f. Thus, all infeasible cells generated by the application of the mutation operators for EC_f: 1) replace the cells with the highest sums of constraint violations from EC_inf if they are better or 2) complete EC_inf, if the size of EC_inf is less than its maximum (predefined) value.

Once the EC_f and EC_inf populations have reacted during a (predetermined) number of times (rep_EC) the best solutions from these populations are inserted (if MC is empty) or the solutions replace the worst solutions in MC (if MC has solutions). First, the cells from EC_f are considered, and then, the cells from EC_inf are considered, if deemed necessary, until the maximum allowable size of MC is reached.

Next, the cells from MC are mutated and evaluated a fixed number of times (rep_MC).

Every 5 iterations (from the outer loop), VC's size is reduced by half and the number of cells to be replaced in EC_f and EC_inf is reduced by two. This aims to provide enough diversity at the beginning of the search as well as to retain the best solutions in the ECs by the end of the search.

The algorithm finishes when a fixed number of evaluations of the objective function is reached.

The general structure of our proposed T-Cell approach is shown in Algorithm 1.

The most relevant aspects of the T-Cell model are the following:

- The fitness of a cell is determined by the objective function value (if the cell is feasible) or by the sum of constraints violation (if the cell is infeasible).
- All the equality constraints are transformed into inequality constraints, using a tolerance factor δ, $|h(\vec{x})| - \delta \leq 0$, with $\delta = 0.0001$.
- VC and MC are sorted using the following criterion: the feasible cells whose objective function values are the best are placed first. Then, we place the infeasible cells that have the lowest sum of constraint violation.
- EC_f are sorted in ascending order based on their objective function values.
- EC_inf are sorted in ascending order based on their sums of constraint violation.
- The required parameters for T-Cell are the following:

    1. **Size of VC:** This is the size of the population of virgin cells. We tested the algorithm with 10, 50 and 100 cells. In general, the best results were obtained with 100 cells.
    2. **Size of the EC_f population:** This is the size of the population of effector feasible cells. We tested the algorithm with 5 and 20 cells. In general, the best results were obtained with 20 cells.

Algorithm 1 Pseudocode of our proposed T-Cell Algorithm

```
1: while the maximum number of evaluations has not been reached do
2:    Randomly generate Virgin Cells (VC)
3:    Evaluate VC
4:    Divide VC into feasible and infeasible cells
5:    Replace a percentage of Effector Cells (EC) with cells from VC
6:    while rep_EC has not been reached do
7:       Make the Effector Cells in EC_inf react by generating EC_inf_new
8:       Evaluate EC_inf_new
9:       Replace cells in EC_inf by their corresponding mutated cells in
         EC_inf_new, only if the mutated cells are better
10:      Sort EC_inf_new with respect to the sum of constraints violation
11:      Insert feasible cells from EC_inf_new into EC_f if required
12:      Make the Effector Cells in EC_f react by generating EC_f_new
13:      Evaluate EC_f_new
14:      Replace cells in EC_f by their corresponding mutated cells in EC_f_new
         only if the mutated cells are better
15:      Sort EC_f_new with respect to the objective function values
16:      Insert infeasible cells from EC_f_new into EC_inf if required
17:   end while
18:   Replace a percentage of Memory Cells with Effector Cells
19:   while rep_MC has not been reached do
20:      Make the Memory Cells (MC) react
21:      Evaluate MC
22:   end while
23:   Decrement the number of cells in VC and EC
24:   Compute Statistics
25: end while
26: Report Statistics
```

3. **Size of the EC_inf population:** This is the size of the population of effector infeasible cells. We tested the algorithm with 5 and 20 cells. In general, the best results were obtained with 20 cells.
4. **Size of the MC population:** This is the size of the population of memory cells. We tested the algorithm with 5, 10 and 20 cells. In general, the best results were obtained with 20 cells.
5. **Number of repetitions for EC (rep_EC):** This is the number of times the cells from EC_f and EC_inf react, are evaluated and selected to pass to the next iteration. We tested the algorithm with values between 10 and 100. In general, the best results were obtained with 100 repetitions.
6. **Number of repetitions for MC (rep_MC):** This is the number of times the cells from MC react, are evaluated and selected to pass to the next iteration. We tested the algorithm with values between

10 and 100. In general, the best results were obtained with 100 repetitions.

7. **Percentage of replacement for EC:** This is the percentage of cells of EC_f and EC_inf that are replaced by cells from VC. Here, we recommend a 100% replacement policy.
8. **Percentage of replacement for MC:** This is the percentage of cells of MC that are replaced by cells from EC_f (and EC_inf if it is necessary). Here, we recommend a 50% replacement policy.
9. **Probability of mutation ($prob_{mut}$):** This is the probability for the mutation operators applied to the effector cells. We tested the algorithm with 0.01, 0.1 and 0.5. In general, the best results were obtained with 0.1.

6 Numerical Experiments

In order to validate our proposed T-Cell algorithm we adopted: 1) a benchmark of 20 test functions taken from the specialized literature (these problems are described in [36] and [28]) and 2) seven engineering optimization problems (described in the Appendix at the end of this paper) which have been previously tackled using metaheuristics.

For all the experiments, we adopted the following parameters, which were empirically derived alter numerous experiments:

1. **Size of VC:** 100.
2. **Size of the EC_f population:** 20
3. **Size of the EC_inf population:** 20
4. **Size of the MC population:** 20
5. **Number of repetitions for EC (rep_EC):** 100
6. **Number of repetitions for MC (rep_MC):** 100
7. **Percentage of replacement for EC:** 100%
8. **Percentage of replacement for MC:** 50%
9. **Probability of mutation ($prob_{mut}$):** 0.1

30 and 50 independents runs were performed for the benchmark and the engineering problems, respectively. All the statistical measures reported were taken only with respect to the runs in which a feasible solution was reached at the end.

Function	BKS	Best	Worst	Mean	Std.Dev
g01	-15.0	**-15.0**	**-15.0**	**-15.0**	0.0
g02	-0.803619	-0.801367	-0.687827	-0.752975	0.032095
g03	-1.0	**-1.0**	**-1.0**	**-1.0**	0.0
g04	-30665.5386	-30665.5385	-30665.5382	-30665.5384	0.0001
g05*	5126.4967	5126.6255	6112.1181	5378.2678	298.0173
g06	-6961.81387	**-6961.81387**	-6961.81365	-6961.81386	0.000039
g07	24.3062	24.3209	25.1347	24.6534	0.219815
g08	-0.095825	**-0.095825**	**-0.095825**	**-0.095825**	0.0
g09	680.63	**680.63**	680.70	680.65	0.0167
g10	7049.24	7050.8342	9054.2923	8020.7551	621.7231
g11	0.7499	**0.7499**	**0.7499**	**0.7499**	0.0
g12	-1.0	**-1.0**	**-1.0**	**-1.0**	0.0
g13	0.0539498	0.054638	0.994983	0.458857	0.344995
g14	-47.7648	-47.5171	-43.272382	-45.3108	1.1156
g15	961.71502	**961.71502**	970.59467	963.37482	2.27562
g16	-1.905155	**-1.905155**	**-1.905155**	**-1.905155**	0.0
g17 **	8853.539	8861.821	9231.201	8990.997	106.193174
g18	-0.86602	-0.86596	-0.62977	-0.78455	0.09746
g19	32.655	33.39078	48.48763	38.92761	3.19102
g20	-5.508013	**-5.508013**	**-5.508013**	**-5.508013**	0.0

Table 1: Results obtained by our proposed T-Cell Algorithm for 20 test problems previously reported in the specialized literature. Our proposed approach performed 350,000 objective function evaluations in all cases. The asterisk (*) indicates a case in which only 24 of the 30 runs converged to a feasible solution. The double asterisk (**) indicates a case in which only 29 of the 30 runs converged to a feasible solution. **BKS** refers to either the global optimum or the best known solution. We show in **boldface** the results that match the optimum value.

6.1 Analysis of Results for the Benchmark Problems

The test functions g02, g03, g08 and g12 are maximization problems (for simplicity, these problems were transformed into minimization problems using $-f(x)$) and the rest are minimization problems. It is worth noting that the test problems adopted are an extended version of the set originally proposed in [31] (which contained 12 test problems) and extended (with one more problem) in [36]. The 7 additional test problems adopted were originally introduced in [28]. These test problems contain characteristics that are representative of what can be considered "difficult" global optimization problems for an evolutionary algorithm. For all these benchmark problems, we adopted a binary representation with Gray codes (for VCs and ECs) with the amount of bits required to obtain an accuracy of 10 digits after the decimal point. All the approaches performed 350,000 objective function evaluations in all cases.

Our results, over the benchmark, are compared with respect to: 1) Stochastic Ranking [36] (the results shown are those reported in [12], which correspond to 19 of the 20 test problems adopted here), 2) an AIS approach for solving con-

Function	BKS	Best	Worst	Mean	Std.Dev
g01	-15.0	**-15.0**	**-15.0**	**-15.0**	-
g02	-0.803619	-0.803	-0.734	-0.784	-
g03	-1.0	**-1.0**	**-1.0**	**-1.0**	-
g04	-30665.5386	**-30665.539**	-30664.216	-30665.480	-
g05	5126.4967	**5126.497**	5153.757	5130.752	-
g06	-6961.81387	**-6961.814**	-6267.787	-6863.645	-
g07	24.3062	24.310	24.830	24.417	-
g08	-0.095825	**-0.095825**	**-0.095825**	**-0.095825**	-
g09	680.63	**680.63**	680.697	680.646	-
g10	7049.24	7050.194	8867.844	7423.434	-
g11	0.7499	0.750	0.751	0.750	-
g12	-1.0	**-1.0**	**-1.0**	**-1.0**	-
g13	0.0539498	0.053	0.128	0.061	-
g14	-47.7648	-41.551	-40.125	-41.551	-
g15	961.71502	961.715	962.008	961.731	-
g16	-1.905155	-1.905	-1.587	-1.703	-
g17	8853.539	8811.692	8559.613	8805.99	-
g18	-0.86602	-0.86600	-0.45700	-0.78600	-
g19	32.655	33.147	37.477	34.337	-

Table 2: Results obtained with Stochastic Ranking [36] in all, but one of the 20 test problems previously indicated. These results were reported in [12], since the original source of the algorithm (see [36]) only reports results for the first 13 test problems (g01 to g13). Stochastic Ranking performed, in all cases, 350,000 objective function evaluations. **BKS** refers to either the global optimum or the best known solution. We show in **boldface** the results that match the optimum value. - indicates that this value is not reported by the authors.

strained problems, which is reported in [14] (this approach was only tested with the first 13 test functions from the set adopted here) and 3) an AIS-GA approach for solving constrained problems, which is reported in [9] (this approach was only tested with the first 13 test functions from the set adopted here). Stochastic ranking uses a multi-membered evolution strategy, gaussian mutation, and real-numbers encoding. It is a very powerful algorithm for constrained optimization, which is frequently adopted for benchmarking new constraint-handling techniques. The approach reported in [14] is one of the few AIS algorithms reported in the specialized literature, which was specifically designed to solve constrained optimization problems. Furthermore, to the authors' best knowledge, it is the most powerful AIS-based approach that has been published so far, for solving constrained optimization problems. The algorithm presented in [9] was chosen in order to have an extra AIS-based approach to compare the performance of our approach.

The results obtained by our proposed approach are shown in Table 1. The results obtained by Stochastic Ranking [36] are shown in Table 2. The results obtained by the AIS approach reported in [14] are shown in Table 3. The results obtained by the AIS approach reported in [9] are shown in Table 4.

Function	BKS	Best	Worst	Mean	Std.Dev
g01	-15.0	-14.9874	-12.9171	-14.7264	0.6070
g02	-0.803619	-0.8017	-0.6268	-0.7434	0.0414
g03	-1.0	**-1.0**	**-1.0**	**-1.0**	0.0
g04	-30665.5386	**-30665.5387**	-30665.5386	-30665.5386	0.0000
g05*	5126.4967	5126.9990	6111.1714	5436.1278	300.88
g06	-6961.81387	-6961.8105	-6961.7981	-6961.8065	0.0027
g07	24.3062	24.5059	26.4223	25.4167	0.4637
g08	-0.095825	**-0.095825**	**-0.095825**	**-0.095825**	0.0000
g09	680.63	680.6309	680.6965	680.6521	0.0176
g10	7049.24	7127.9502	12155.1358	8453.7902	1231.37
g11	0.7499	0.75	0.75	0.75	0.0000
g12	-1.0	**-1.0**	**-1.0**	**-1.0**	0.0000
g13	0.0539498	0.05466	1.49449	0.45782	0.3790

Table 3: Results obtained by the AIS proposed in [14]. The asterisk (*) indicates a case in which only 90% of the runs converged to a feasible solution. It is worth noting that the authors report results only for the first 13 test functions from the benchmark adopted here. This approach also performed 350,000 objective function evaluations. **BKS** refers to either the global optimum or the best known solution. We show in **boldface** the results that match the optimum value.

Function	BKS	Best	Worst	Mean	Std.Dev
g01	-15.0	-14.9944	-14.9687	-14.9793	-
g02	0.803619	0.772831	0.756478	0.764654	-
g03	1.0	0.9989538	0.9933335	0.9978105	-
g04	30655.5386	30665.53	30665.25	30665.35	-
g05	5126.4967	INF	INF	INF	-
g06	6961.81387	6961.804	6961.804	6961.804	-
g07	24.3062	25.373074	26.896858	25.888315	-
g08	-0.095825	**-0.095825**	**-0.095825**	**-0.095825**	-
g09	680.63	680.6817	681.0401	680.7827	-
g10	7049.24	7320.2637	8081.6685	7571.3228	-
g11	0.7499	0.750035	0.9992529	0.878316	-
g12	-1.0	**-1.0**	**-1.0**	**-1.0**	-
g13	0.0539498	INF	INF	INF	-

Table 4: Results obtained by the AIS proposed in [9]. INF means that the algorithm converged to an infeasible solution. It is worth noting that the authors report results only for the first 13 test functions from the benchmark adopted here. This approach performs 350,000 objective function evaluations. **BKS** refers to either the global optimum or the best known solution. We show in **boldface** the results that match the optimum value. - indicates that this value is not reported by the authors.

From Table 1, we can see that our proposed approach was able to reach the global optimum in 10 test functions (g01, g03, g06, g08, g09, g11, g12, g15, g16 and g20). Additionally, our approach reached feasible solutions close to the global optimum (in the best case) in the rest of the test functions, showing its worst performance in g05, in which it reached the feasible region only in 24 of the 30 runs performed. Comparing the performance of our proposed approach with respect to Stochastic Ranking (see Tables 1 and 2), we can see that our approach obtained better results in 5 test functions (g11, g14, g15, g16 and g17). Both approaches found similar solutions in six other test problems: g01, g03, g06, g08, g09 and g12. Our proposed model was outperformed in the rest of the problems. Considering average results, our approach was outperformed by stochastic ranking in 7 problems: g02, g05, g09, g10, g13, g15, g19. Thus, from this comparison, we can conclude that our T-Cell approach is very competitive with respect to Stochastic Ranking.

Comparing our proposed T-Cell approach with respect to the AIS proposed in [14] (see Tables 1 and 3), our approach obtained better best results in 8 test functions (g01, g05, g06, g07, g09, g10, g11 and g13). Both approaches found similar best results for g08 and g12. Our proposed model was outperformed (in terms of best results found) in g04. With respect to the mean and worst found solutions, our approach was outperformed only in g02, g04 and g09. Thus, we can conclude that our proposed T-Cell outperforms the AIS proposed in [14].

Comparing our proposed T-Cell approach with respect to the AIS proposed in [9] (see Tables 1 and 4), our approach obtained better best results in all test functions. Both approaches found similar best results for g08 and g12. With respect to the mean and worst found solutions, our approach was outperformed only in g02 and g10. Thus, we can conclude that our proposed T-Cell outperforms the AIS proposed in [9]. It is worth noting, however, that our proposed approach adopts a hybrid encoding scheme that incorporates both binary and real numbers encoding, whereas the AIS proposed in [9] only adopts binary encoding.

6.2 Analysis of Results for the Engineering Optimization Problems

Seven mechanical engineering optimization problems (the description of these problems is available at the Appendix included at the end of this paper) were used to validate the performance of our approach.

In these cases, our results are compared with respect to several bio-inspired metaheuristics, which have reported the best values known so far for the engineering problems adopted in our comparative study.

The problems Tension/Compression Spring Design, Speed Reducer Design, Welded Beam Design, Pressure Vessel Design, and 10-bar plane truss are compared with respect to the following approaches: three genetic algorithms hybridized with an artificial immune system (AIS-GA) [10] and its modified version AIS-GA^C which includes a clearing procedure [10] and another version

presented in [8]; a binary coded genetic algorithm equipped with the adaptive penalty method (APM) [6], and the Stochastic Ranking technique [36].

The number of objective function evaluations performed for each engineering optimization problem is different, since we adopt the values reported by the other authors, in order to perform a fair comparison. For the Tension/Compression Spring Design and the Speed Reducer Design problems all the approaches performed 36,000 evaluations. For the Welded Beam Design problem all the approaches performed 320,000 evaluations. For the Pressure Vessel Design problem all the approaches performed 80,000 evaluations of the objective function and for the 10-bar plane truss problem, all the approaches performed 280,000 evaluations. For this last problem we just compared results with respect to the continuous case. All the approaches, including our own, used binary representation with Gray codes (for VC and EC), with strings of 25 bits in length to encode each design variable. For 10-bar plane truss, 25-bar space truss and 200-bar plane truss all design variables are assumed to be involved in all the constraints, when the first mutation operator for EC_f is applied.

Table 5 shows the comparison of results for the Tension/Compression Spring Design problem. The results from Table 5 clearly indicate that our proposed approach outperforms all the other approaches with respect to which it was compared. All approaches found feasible solutions in the 50 runs performed. The AIS-GAs from [10] do not report this value. Table 6 shows the decision variables corresponding to the best solutions found by each approach for this problem. All the decision variables correspond to feasible solutions. The constraint values for the best solution found by our T-Cell are: $g_1 = -0.000029$, $g_2 = -0.000004$, $g_3 = -4.050423$ and $g_4 = -0.728849$.

Algorithm	**Best**	**Worst**	**Mean**	**St. Dev**
T-Cell	**0.012665**	**0.013309**	**0.012732**	9.4E-5
AIS-GA [10]	0.012668	0.016155	0.013481	-
AIS-GA^C [10]	0.012666	0.013880	0.012974	-
AIS-GA [8]	0.012666	0.015318	0.013131	6.28E-4
APM [5]	0.012684	0.017794	0.014022	1.47E-3
SR [36]	0.012679	0.017796	0.013993	1.27-3

Table 5: Best objective function found for the Tension/Compression Spring Design problem. The number of function evaluations performed by all approaches was 36,000. - indicates that this value is not reported by the authors.

Table 7 shows the comparison of results for the Speed Reducer Design problem. The results from Table 7 indicate that our proposed approach found the best solution with respect to which it was compared, but T-Cell only found feasible solutions in 36 out of the 50 runs performed. APM [5] found feasible solutions in 19 out of the 50 runs performed, and the remaining approaches found feasible solutions in the 50 runs. Table 8 shows the decision variables corresponding to the best solution found by each approach for this problem. All the decision variables shown correspond to feasible solutions. The constraint values for the

	T-Cell	AIS-GA [10]	AIS-GA^C [10]	AIS-GA [8]	APM [5]	SR [36]
x_1	11.384534	11.852177	11.329555	11.6611924	12.070748	11.375795
x_2	0.355105	0.347475	0.356032	0.3505298	0.344304	0.355485
x_3	0.051622	0.051302	0.051661	0.0514305	0.051168	0.051638
V	**0.012665**	0.012668	0.012666	0.012666	0.0126838	0.012679

Table 6: Design variables corresponding to the best solutions found for the Tension/Compression Spring Design problem.

best solution found by our T-Cell are: $g_1 = -0.073915$, $g_2 = -0.197999$, $g_3 = -0.499172$, $g_4 = -0.901472$, $g_5 = -0.000000$, $g_6 = 0.000000$, $g_7 = -0.702500$, $g_8 = -0.000000$, $g_9 = -0.583333$, $g_{10} = -0.051326$ and $g_{11} = -0.010852$.

Algorithm	**Best**	**Worst**	**Mean**	**St. Dev**
T-Cell	**2996.3481**	2996.3801	2996.3551	8.9E-3
AIS-GA [10]	2996.3494	2996.6277	2996.3643	4.35E-3
AIS-GA^C [10]	2996.3484	**2996.3486**	**2996.3484**	1.46E-6
AIS-GA [8]	2996.3483	2996.3599	2996.3501	7.45E-3
APM [5]	2996.3482	3459.0948	3033.8807	1.10E-2
SR [36]	2996.3483	2996.3535	2996.3491	1.01E-3

Table 7: Best objective function values found for the Speed Reducer Design problem. The number of function evaluations performed by all the algorithms was 36,000.

	T-Cell	AIS-GA [10]	AIS-GA^C [10]	AIS-GA [8]	APM [5]	SR [36]
x_1	3.500000	3.500001	3.500000	3.500001	3.500000	3.500000
x_2	0.700000	0.700000	0.700000	0.700000	0.700000	0.700000
x_3	17	17	17	17	17	17
x_4	7.300000	7.300019	7.300001	7.300008	7.300000	7.300001
x_5	7.800000	7.800013	7.800000	7.800001	7.800000	7.800001
x_6	3.350215	3.350215	3.350215	3.350215	3.350215	3.350215
x_7	5.286683	5.286684	5.286684	5.286683	5.286683	5.286683
W_1	**2996.3481**	2996.3494	2996.3484	2996.3483	2996.3482	2996.3483

Table 8: Design variables corresponding to the best solutions found for the Speed Reducer Design problem.

Table 9 shows the comparison of results for the Welded Beam Design problem. The results from Table 9 clearly indicate that our proposed approach outperforms all the other approaches with respect to the best found solution, but not with respect to the Worst and Mean values (AIS-GA^C [10] was better in that regard). All the approaches found feasible solutions in all runs.[4] Table 10 shows the decision variables corresponding to the best solutions found

[4]For AIS-GA [10] this value is not reported.

by each approach for this problem. All the decision variables shown correspond to feasible solutions. The constraint values for the best solution found by our T-Cell are: $g_1 = -0.642858$, $g_2 = -0.021854$, $g_3 = -0.000000$, $g_4 = -0.004550$ and $g_5 = -0.234241$.

Algorithm	**Best**	**Worst**	**Mean**	**St. Dev**
T-Cell	**2.38113**	2.710406	2.439811	0.093146
AIS-GA [10]	2.38125	3.23815	2.59303	-
AIS-GA^C [10]	2.38122	**2.41391**	**2.38992**	-
AIS-GA [8]	2.38335	4.05600	2.992998	2.02E-1
APM [5]	2.38144	5.94803	3.49560	9.09E-1
SR [36]	2.59610	10.1833	4.33259	1.29

Table 9: Best objective function values found for the Welded Beam Design problem. The number of function evaluations performed by all the algorithms was 320,000. - indicates that this value is not reported by the authors.

	T-Cell	AIS-GA [10]	AIS-GA^C [10]	AIS-GA [8]	APM [5]	SR [36]
h	0.244369	0.2443243	0.2443857	0.2434673	0.2442419	0.2758192
l	6.218613	6.2201996	6.2183037	6.2507296	6.2231189	5.0052613
t	8.291474	8.2914640	8.2911650	8.2914724	8.2914718	8.6261101
b	0.244369	0.2443694	0.2443875	0.2443690	0.2443690	0.2758194
C	**2.38113**	2.381246	2.38122	2.38335	2.38144	2.59610

Table 10: Design variables corresponding to the best solutions found for the Welded Beam Design problem.

Table 11 shows the comparison of results for the Pressure Vessel Design problem. The results from Table 11 clearly indicate that our proposed approach is outperformed by all the other approaches with respect to which it was compared, except for SR, when we take into account only the best solution found. All the approaches found feasible solutions in all runs.[5] Table 12 shows the decision variables corresponding to the best solutions found by each approach for this problem. All the decision variables shown correspond to feasible solutions. The constraint values for the best solution found by our T-Cell are: $g_1 = -0.000000$, $g_2 = -0.035881$, $g_3 = -0.114149$ and $g_4 = -430.787695$.

Table 13 shows the comparison of results for the 10-bar plane truss. The results from Table 13 clearly indicate that our proposed approach was outperformed by all the other approaches with respect to which it was compared. All the approaches found feasible solutions in all runs.[6] Table 14 shows the decision variables corresponding to the best solutions found by each approach for this problem. All the decision variables shown correspond to feasible solutions.

The results of the 25-bar space truss problem were compared with respect to a particle swarm optimizer coupled with an augmented Lagrange multiplier

[5]For AIS-GA [10] this value is not reported.

[6]For AIS-GA [10] this value is not reported.

Algorithm	**Best**	**Worst**	**Mean**	**St. Dev**
T-Cell	6390.554	7694.066881	6737.065147	3.57E+2
AIS-GA [10]	6060.368	7546.750	6743.872	-
AIS-GA^C [10]	6060.138	**6845.496**	**6385.942**	-
AIS-GA [8]	**6059.855**	7388.160	6545.126	1.24E+2
APM [5]	6065.822	8248.003	6632.376	5.15E+2
SR [36]	6832.584	8012.615	7187.314	2.67E+2

Table 11: Best objective function values found for the Pressure Vessel Design problem. The number of function evaluations performed by all the algorithms was 80,000.

	T-Cell	AIS-GA [10]	AIS-GA^C [10]	AIS-GA [8]	APM [5]	SR [36]
T_s	0.8125	0.8125	0.8125	0.8125	0.8125	1.1250
T_h	0.4375	0.4375	0.4375	0.4375	0.4375	0.5625
R	42.098429	42.0931	42.0950	42.0973	42.0492	58.1267
L	190.787695	176.7031	176.6797	176.6509	177.2522	44.5941
W_2	6390.554	6060.367	6060.138	**6059.854**	6065.821	6832.583

Table 12: Design variables corresponding to the best solutions found for the Pressure Vessel Design problem.

formulation and a dynamic inertia weight variation method (PSO) [35]. We did not find in the specialized literature any AIS which had adopted this problem. Table 15 shows the results found by our T-Cell and the previously indicated PSO. T-Cell performed 50 runs with 2,000 objective function evaluations per run. The authors of the PSO approach do not report the number of objective function evaluations performed, but they indicate that they performed 20 independent runs.

T-Cell found feasible solutions in all runs. In [35]. this value is not reported. Table 16 shows the decision variables corresponding to the best solutions found by these approaches. All the decision variables shown correspond to feasible solutions.

Finally, the comparison of results for the 200-bar plane truss is shown in Table 17. In this case, we did not find an AIS approach with respect to which we could compare our results and, therefore, we decided to compare results with respect to the following methods from [7]: feasible directions (CONMIN), Pshenichny's Recursive Quadratic Programming (LINRM), gradient projection (GPR-UI), exterior penalty function (SUMT), multiplier methods (M-3, M-4 and M-5). Additionally, we compared results with respect to the following approaches: a harmony search algorithm [27], and a trust region method for structural optimization which uses exact second order sensitivity (TRUST) [38].

The harmony search algorithm reported in [27] performed 50,000 objective function evaluations. The other approaches do not report the number of objective function evaluations performed. Based on the values adopted by the

Algorithm	**Best**	**Worst**	**Mean**	**Std.Dev**
T-Cell	5142.30	5164.2821	5148.3626	4.90
AIS-GA [10]	5062.67	5094.8867	5075.5513	-
AIS-GA^C [10]	5064.67	5113.22	5082.52	-
AIS-GA [8]	**5061.16**	**5084.56**	**5068.85**	7.78
APM [8]	5062.12	6430.55	5133.22	2.48E+2
SR [36]	5061.71	5101.17	5077.67	1.01E+1

Table 13: Best objective function values found for the 10-bar plane truss problem. The number of function evaluations performed by all the algorithms was 280,000. - indicates that this value is not reported by the authors.

	T-Cell	AIS-GA [10]	AIS-GA^C [10]	AIS-GA [8]	APM [8]	SR [36]
x_1	31.23829	30.16252	29.78121	30.52684	30.95080	30.01400
x_2	0.316625	0.10004	0.10031	0.10000	0.10000	0.10000
x_3	23.61073	22.81192	22.55140	22.91574	22.92083	26.14460
x_4	14.50669	15.87183	15.50462	15.48294	15.55024	15.29260
x_5	0.316234	0.10000	0.10002	0.10000	0.10000	0.10000
x_6	0.316464	0.51495	0.52377	0.54620	0.60959	0.55610
x_7	8.135098	7.50595	7.52854	7.47594	7.46973	7.43980
x_8	21.61828	21.26408	21.15708	21.01566	20.83562	21.00560
x_9	21.22159	21.38304	22.21351	21.55362	21.35644	21.93900
x_{10}	0.31634	0.10001	0.10018	0.10000	0.10000	0.10000
W_3	5142.30	5062.67	5064.67	**5061.16**	5062.12	5061.71

Table 14: Design variables corresponding to the best solutions found by each approach for the 10-bar plane truss.

other metaheuristic with respect to which we compared our results (i.e., harmony search), we performed 30 independent runs with 50,000 objective function evaluations per run.

Here, we adopted the following parameters in order to perform 50,000 objective function evaluations (other parameters had to be empirically tuned in this case, considering the lower population size that had to be adopted for VC):

1. **Size of VC:** 50.
2. **Size of the EC_f population:** 20
3. **Size of the EC_inf population:** 20
4. **Size of the MC population:** 10
5. **Number of repetitions for EC (rep_EC):** 10
6. **Number of repetitions for MC (rep_MC):** 10
7. **Percentage of replacement for EC:** 100%
8. **Percentage of replacement for MC:** 50%

Algorithm	Best	Worst	Mean	Std.Dev
T-Cell	**471.332**	504.502	479.205	6.17
PSO [35]	483.84	**489.424**	-	-

Table 15: Best objective function values found for the 25-bar space truss. - indicates that this value is not reported by the authors.

	T-Cell	PSO [35]
x_1	0.171600	0.1000
x_2	0.215618	0.4565
x_3	3.530105	3.4000
x_4	0.179024	0.1000
x_5	1.968327	1.9369
x_6	0.862956	0.9647
x_7	0.152496	0.4423
x_8	3.787504	3.4000
W_4	**471.332**	483.84

Table 16: Design variables corresponding to the best solutions found by all the approaches for the 25-bar space truss.

9. **Probability of mutation ($prob_{mut}$):** 0.01

Our proposed approach outperformed all the approaches with respect to which it was compared. T-Cell found feasible solutions in all the runs performed. Table 18 shows the decision variables found by T-Cell and the harmony search algorithm reported in [27]. All the decision variables shown correspond to feasible solutions. This information was not available for any of the other approaches.

Algorithm	Best	Worst	Mean	Std.Dev
T-Cell	**24852.58**	33132.30	27376.57	2165.0667
CONMIN	34800.0	-	-	-
LINRM	33315.0	-	-	-
SUMT	27564.0	-	-	-
M-3	26600.0	-	-	-
M-4	26654.0	-	-	-
M-5	26262.0	-	-	-
Harmony search [27]	25447.10	-	-	-
TRUST [38]	25500.8	-	-	-

Table 17: Best objective function values found for the 200-bar plane truss. - indicates that this value is not reported by the authors.

	T-Cell	Harmony search [27]
x_1	0.5077	0.1253
x_2	0.9850	1.0157
x_3	0.6700	0.1069
x_4	0.4076	0.1096
x_5	2.5113	1.9369
x_6	0.3334	0.2686
x_7	0.6602	0.1042
x_8	3.7534	2.9731
x_9	0.4640	0.1309
x_{10}	4.3729	4.1831
x_{11}	0.4282	0.3967
x_{12}	0.3472	0.4416
x_{13}	5.0191	5.1873
x_{14}	0.4378	0.1912
x_{15}	6.3160	6.2410
x_{16}	0.4859	0.6994
x_{17}	0.5247	0.1158
x_{18}	7.4365	7.7643
x_{19}	0.4013	0.1000
x_{20}	7.9161	8.8279
x_{21}	0.8591	0.6986
x_{22}	1.6976	1.5563
x_{23}	10.1246	10.9806
x_{24}	0.4382	0.1317
x_{25}	10.9628	12.1492
x_{26}	1.6424	1.6373
x_{27}	3.8004	5.0032
x_{28}	8.4192	9.3545
x_{29}	13.5780	15.0919
W_5	**24852.58**	25447.10

Table 18: Design variables corresponding to the best solutions found by each approach for the 200-bar plane truss.

7 Conclusions and Future Work

We have presented a modified version of an artificial immune system based on the T-Cell model, which has been proposed to solve constrained optimization problems. The proposed approach is inspired on the processes suffered by the T-Cells within our immune system. The proposed approach has been validated first with a benchmark of standard test functions normally adopted to assess the performance of new constraint-handling techniques coupled to evolutionary algorithms. In this case, our results were compared with respect to those generated by an evolutionary algorithm and two artificial immune systems which are representative of the state-of-the-art in constrained optimization. After that, we also adopted seven engineering optimization problems, and compared results with respect to approaches previously reported in the specialized literature. In both cases, the results obtained by our proposed approach were competitive, resulting better in most cases to those generated by the other algorithms with respect to which it was compared. This indicates that our T-Cell algorithm can

be a viable alternative to solve constrained optimization problems in engineering or other disciplines.

As part of our future work, we are interested in studying mechanisms that allow us to reduce the number of parameters that our approach requires. Such parameters are mainly derived from the biological model that we developed as part of this work. Such model is relatively complex and yet represents only a raw approximation of the real phenomenon that takes place within our immune system. However, we believe that it is possible to introduce further modifications to the model that allow a reduction of its parameters, or the adoption of fixed values for most of them. This would certainly simplify its use. Additionally, we aim to explore techniques that increase the robustness of our proposed approach (i.e., we aim to reduce the standard deviations obtained in certain cases and to improve our results in those cases in which other approaches outperformed ours).

Acknowledgments

The authors thank the anonymous reviewers for their comments, corrections and suggestions, which greatly helped them to improve the contents of this paper.

The third author acknowledges support from CONACyT project no. 103570.

Appendix: Mechanical Engineering Problems

This appendix describes seven engineering problems that were used to test our proposed T-Cell Algorithm.

1. **The Tension/Compression Spring Design**

   The objective is to minimize the volume V of a coil spring under a constant tension/compression load. The design variables are the number of active coils of the spring ($N = x_1 \in [2, 15]$), the winding diameter ($D = x_2 \in [0.25, 1.3]$), and the wire diameter ($d = x_3 \in [0.05, 2]$). The volume and the mechanical constraints are given by:

$$V(x) = (x_1 + 2)x_2x_3^2$$

$$g_1(x) = 1 - \frac{x_2^3 x_1}{71785x_3^4} \leq 0$$

$$g_2(x) = \frac{4x_2^2 - x_3x_2}{12566(x_2x_3^3 - x_3^4)} + \frac{1}{5108x_3^2} - 1 \leq 0$$

$$g_3(x) = 1 - \frac{140.45x_3}{x_2^2x_1} \leq 0$$

$$2\ g_4(x) = \frac{x_2 + x_3}{1.5} - 1 \leq 0$$

2. **The Speed Reducer Design**

The objective is to minimize the weight W of a speed reducer. The design variables are the face width ($b = x_1 \in [2.6, 3.6]$), the module of teeth ($m = x_2 \in [0.7, 0.8]$), the number of teeth on pinion ($n = x_3 \in [17, 28]$), the length of the shaft 1 between the bearings ($l_1 = x_4 \in [7.3, 8.3]$), the length of the shaft 2 between the bearings ($l_2 = x_5 \in [7.8, 8.3]$), the diameter of the shaft 1 ($d_1 = x_6 \in [2.9, 3.9]$), and, finally, the diameter of the shaft 2 ($d_2 = x_7 \in [5.0, 5.5]$). The variable x_3 is integer and all the others are continuous. The weight and the mechanical constraints are given by

$$W_1(x) = 0.7854x_1x_2^2(3.3333x_3^2 + 14.9334x_343.0934) - 1.508x_1(x_6^2 + x_7^2) + 7.4777(x_6^3 + x_7^3) + 0.7854(x_4x_6^2 + x_5x_7^2)$$

$$g_1(x) = \frac{27}{x_1x_2^2x_3} - 1 \leq 0$$

$$g_2(x) = \frac{397.5}{x_1x_2^2x_3^2} - 1 \leq 0$$

$$g_3(x) = \frac{1.93x_4^3}{x_2x_3x_6^4} - 1 \leq 0$$

$$g_4(x) = \frac{1.93x_5^3}{x_2x_3x_7^4} - 1 \leq 0$$

$$g_5(x) = \frac{\left[\left(\frac{745x_4}{x_2x_3}\right)^2 + 16.9\times10^6\right]^{\frac{1}{2}}}{110.0x_6^3} - 1 \leq 0$$

$$g_6(x) = \frac{\left[\left(\frac{745x_5}{x_2x_3}\right)^2 + 157.5\times10^6\right]^{\frac{1}{2}}}{85.0x_7^3} - 1 \leq 0$$

$$g_7(x) = \frac{x_2x_3}{40} - 1 \leq 0$$

$$g_8(x) = \frac{5x_2}{x_1} - 1 \leq 0$$

$$g_9(x) = \frac{x_1}{12x_2} - 1 \leq 0$$

$$g_{10}(x) = \frac{1.5x_6 + 1.9}{x_4} - 1 \leq 0$$

$$g_{11}(x) = \frac{1.1x_7 + 1.9}{x_5} - 1 \leq 0$$

3. **The Welded Beam design**

The objective is to minimize the cost $C(h, l, t, b)$ of the beam where $h \in [0.125, 10]$, and $0.1 \leq l, t, b \leq 10$. The objective and constraints are:

$$C(h,l,t,b) = 1.10471h^2l + 0.04811tb(14.0+l)$$

$$g_1(\tau') = -13600 + \sqrt{\tau'^2 + \tau''^2 + l\tau'\tau''/\alpha} \leq 0$$

$$g_2(\sigma) = -30000 + \frac{504000}{t^2b} \leq 0$$

$$g_3(b,h) = -b + h \leq 0$$

$$g_4(Pc) = -Pc + 6000 \leq 0$$

$$g_5(t,b) = -0.25 + \frac{2.1952}{t^3b} \leq 0$$

$$\tau' = \frac{6000}{\sqrt{2}hl}$$

$$\alpha = \sqrt{0.25(l^2 + (h+t)^2)}$$

$$Pc = 64746.022(1 - 0.0282346t)tb^3$$

$$\tau'' = \frac{6000(14+0.5l)\alpha}{2\Big(0.707hl\Big(\frac{l^2}{12}+0.25(h+t)^2\Big)\Big)}$$

4. **The Pressure Vessel Design**

   This problem corresponds to the weight minimization of a cylindrical pressure vessel with two spherical heads. There are four design variables (in inches): the thickness of the pressure vessel (T_s), the thickness of the head (T_h), the inner radius of the vessel (R) and the length of the cylindrical component (L). Since there are two discrete variables (T_s and T_h) and two continuous variables (R and L), one has a nonlinearly constrained mixed discrete-continuous optimization problem. The bounds of the design variables are $0.0625 \leq T_s$, $T_h \leq 5$ (in constant steps of 0.0625) and $10 \leq R$, $L \leq 200$. The weight, to be minimized, and the constraints are given by:

$$W_2(T_s,T_h,R,L) = 0.6224T_sRL + +1.7781T_hR^2 + 3.1661T_s^2L + 19.84T_s^2R$$

$$g_1(T_s,R) = -T_s + 0.0193R \leq 0$$

$$g_2(T_h,R) = -Th + 0.00954R \leq 0$$

$$g_3(R,L) = -\pi R^2L - 4/3\pi R^3 + 1296000 \leq 0$$

$$g_4(L) = L - 240 \leq 0$$

5. **10-Bar Plane Truss**

The geometry of the 10-bar plane truss structure employed in this problem is shown in Figure 1. The problem is to find the cross-sectional area of each member of this truss, such that we minimize its weight (W_3). The problem is subject to both displacement and stress constraints. The weight of the truss is given by

$$f(x) = W = \sum_{j=1}^{n} \rho A_j L_j \tag{8}$$

where x is the candidate solution, A_j is the cross-sectional area of the jth member, L_j is the length of the jth member, and ρ is the weight density of the material. The assumed data are: Young's modulus of elasticity is 10^4 ksi, $\rho = 0.10lb/in^3$ and a load of 100 kips in the negative y-direction is applied at nodes 2 and 4. The maximum allowable stress of each member (σ_a) is assumed to be ± 25 ksi. The maximum allowable displacement of each node (horizontal and vertical)(μ_a) is assumed to be 2 in. The minimum allowable cross-sectional area is 0.10 in^2 for all members. The problem has a total of 10 design variables.

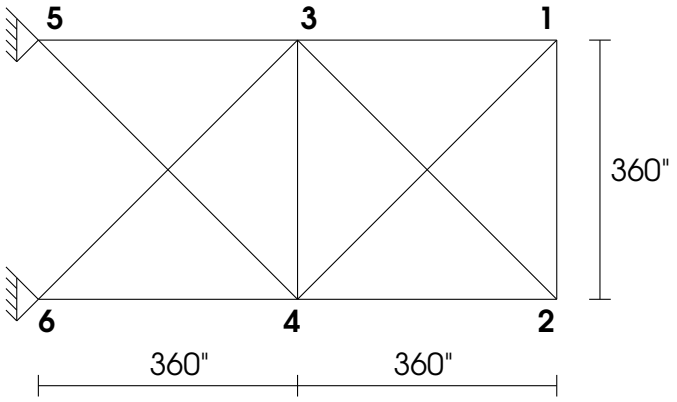

Figure 1: 10-bar plane truss

6. **25-Bar Space Truss**

Figure 2 shows the geometry of this problem, which consists of a 25-bar space truss. The members are divided into eight groups, according to Table 19. The assumed data are: Young's modulus of elasticity 10^7 ksi and specific weight $\rho = 0.10lb/in^3(2,770kg/\ m^3)$, with the applied loads listed in Table 20. Again, the objective function of the problem is set to minimize the weight of the structure W_4. The stress is constrained to

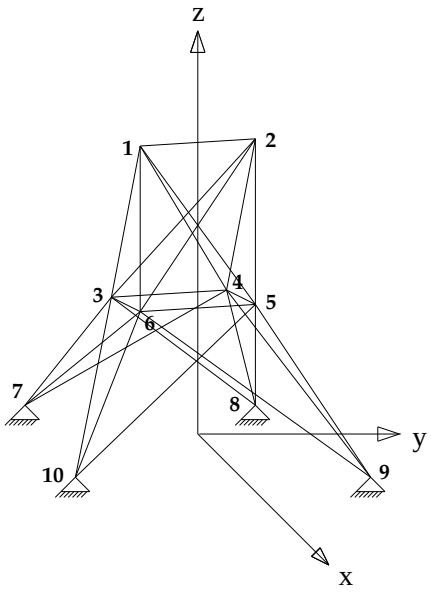

Figure 2: 25-bar space truss

$\pm$ 40 ksi (257.6 MPa) and only the displacements at joints 1 and 2 are restricted, both to less than ± 0.35 in (8.89 mm) in the x and y directions.

7. **200-Bar Plane Truss**

This problem consists of a 200-bar plane truss originally proposed in [7], and shown in Figure 3. The structure has 77 nodes. The problem is to find the cross-sectional area of each member of this truss, such that we minimize its weight W_5. The truss is to be designed under three independent load conditions and is subject only to stress constraints in its members. The three loading conditions are: 1) 1 kip acting in the positive x-direction at node points 1, 6, 15, 20, 29, 34, 43, 48, 57, 62, and 71; 2) 10 kips acting in the negative y-direction at node points 1, 2, 3, 4, 5, 6, 8, 10, 12, 14, 15, 16, 17, 18, 19, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 71, 72, 73, 74, and 75; and 3) load conditions 1 and 2 acting together. The 200 elements of this truss are classified in 29 groups. The grouping information is shown in Table 21. The stress in each element is limited to a value of 10 ksi for both tension and compression members. Young's modulus of elasticity is 30000 ksi and the weight density is 0.283×10^{-3} kips/in^3. A lower bound of 0.1 in^2 and an upper bound of 30 in^2 are imposed on each of the 29

Group Number	Members
1	1-2
2	1-4, 2-3, 1-5, 2-6
3	2-5, 2-4, 1-3, 1-6
4	3-6, 4-5
5	3-4, 5-6
6	3-10, 6-7, 4-9, 5-8
7	3-8, 4-7, 6-9, 5-10
8	3-7, 4-8, 5-9, 6-10

Table 19: Group membership for the 25-bar space truss

Node Number	$F_x(kips)$	$F_y(kips)$	$F_z(kips)$
1	1.00	-10.0	-10.0
2	-	-10.0	-10.0
3	0.5	-	-
6	0.6	-	-

Table 20: Load conditions for 25-bar space truss

decision variables.

References

[1] Victoria S. Aragón, Susana C. Esquivel, and Carlos A. Coello Coello. A Novel Model of Artificial Immune System for Solving Constrained Optimization Problems with Dynamic Tolerance Factor. In Alexander Gelbukh and Ángel Fernando Kuri Morales, editors, *MICAI 2007: Advances in Artificial Intelligence, 6th International Conference on Artificial Intelligence*, pages 19–29, Aguascalientes, México, November 2007. Springer. Lecture Notes in Artificial Intelligence Vol. 4827.

[2] Victoria S. Aragón, Susana C. Esquivel, and Carlos A. Coello Coello. Optimizing Constrained Problems through a T-Cell Artificial Immune System. *Journal of Computer Science & Technology*, 8(3):158–165, 2008.

[3] Victoria S. Aragón, Susana C. Esquivel, and Carlos A. Coello Coello. Solving constrained optimization using a t-cell artificial immune system. *Revista Iberoamericana de Inteligencia Artificial*, 12(40):7–22, 2008.

[4] Jerzy Balicki. Multi-criterion Evolutionary Algorithm with Model of the Immune System to Handle Constraints for Task Assignments. In Leszek Rutkowski, Jörg H. Siekmann, Ryszard Tadeusiewicz, and Lotfi A. Zadeh, editors, *Artificial Intelligence and Soft Computing - ICAISC 2004, 7th International Conference. Proceedings*, pages 394–399, Zakopane, Poland, June 2004. Springer. Lecture Notes in Computer Science. Volume 3070.

Group number	Member number
1	1, 2, 3, 4
2	5, 8, 11, 14, 17
3	19, 20, 21, 22, 23, 24
4	18, 25, 56, 63, 94, 101, 132, 139, 170, 177
5	26, 29, 32, 35, 38
6	6, 7, 9, 10, 12, 13, 15, 16, 27, 28, 30, 31, 33, 34, 36, 37
7	39, 40, 41, 42
8	43, 46, 49, 52, 55
9	57, 58, 59, 60, 61, 62
10	64, 67, 70, 73, 76
11	44, 45, 47, 48, 50, 51, 53, 54, 65, 66, 68, 69, 71, 72, 74, 75
12	77, 78, 79, 80
13	81, 84, 87, 90, 93
14	95, 96, 97, 98, 99, 100
15	102, 105, 108, 111, 114
16	82, 83, 85, 86, 88, 89, 91, 92, 103, 104, 106, 107, 109, 110, 112, 113
17	115, 116, 117, 118
18	119, 122, 125, 128, 131
19	133, 134, 135, 136, 137, 138
20	140, 143, 146, 149, 152
21	120, 121, 123, 124, 126, 127, 129, 130, 141, 142, 144, 145, 147, 148, 150, 151
22	153, 154, 155, 156
23	157, 160, 163, 166, 169
24	171, 172, 173, 174, 175, 176
25	178, 181, 184, 187, 190
26	158, 159, 161, 162, 164, 165, 167, 168, 179, 180, 182, 183, 185, 186, 188, 189
27	191, 192, 193, 194
28	195, 197, 198, 200
29	196, 199

Table 21: Group membership for the 200-bar plane truss

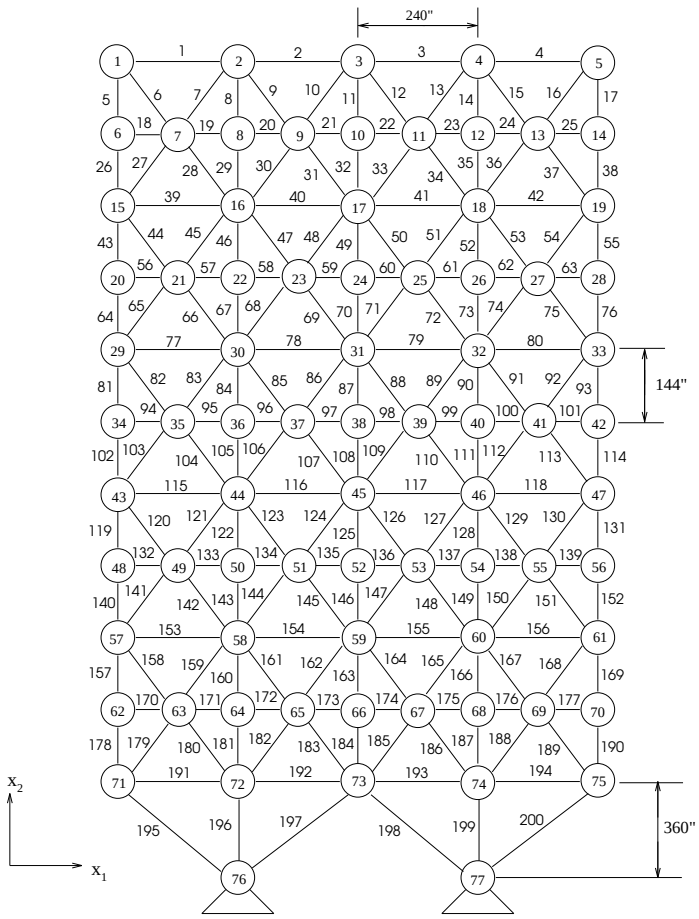

Figure 3: 200-bar plane truss

[5] Helio J.C. Barbosa and Afonso C.C. Lemonge. An adaptive penalty scheme in genetic algorithms for constrained optimization problems. In W.B. Langdon, E.Cantú-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. A. Potter, A.C. Schultz, J. F. Miller, E. Burke, and N.Jonoska, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2002)*, pages 287–294, San Francisco, California, July 2002. Morgan Kaufmann Publishers.

[6] Helio J.C. Barbosa and Alfonso C.C. Lemonge. An adaptive penalty scheme for genetic algorithms in structural optimization. *International Journal for Numerical Methods in Engineering*, 59:703–736, 2004.

[7] A. D. Belegundu. *A study of mathematical programming methods for struc-*

tural optimization. PhD thesis, Department of Civil and Environmental Engineering, USA, 1982.

[8] H. S. Bernardino, H. J. C. Barbosa, A. C. C. Lemonge, and L. G. Fonseca. A New Hybrid AIS-GA for Constrained Optimization Problems in Mechanical Engineering. In *2008 Congress on Evolutionary Computation (CEC'2008)*, pages 1455–1462, Hong Kong, June 2008. IEEE Service Center.

[9] H. S. Bernardino, H. J.C. Barbosa, and A. C.C. Lemonge. Constraints handling in genetic algorithms via artificial immune systems. In *Genetic and Evolutionary Computation - GECCO 2006, Genetic and Evolutionary Computation Conference - Late Breaking Paper*, Seattle, WA, USA, July 2006.

[10] H.S. Bernardino, H.J.C. Barbosa, and A.C.C. Lemonge. A hybrid genetic algorithm for constrained optimization problems in mechanical engineering. In *2007 IEEE Congress on Evolutionary Computation (CEC 2007)*, pages 646–653, Singapore, September 2007. IEEE Press. ISBN: 978-1-4244-1339-3.

[11] P. Bretscher and M. Cohn. A theory of self-nonself discrimination. *Science*, 169:1042–1049, 1970.

[12] Leticia Cagnina, Susana Esquivel, and Carlos Coello-Coello. A Bi-population PSO with a Shake-Mechanism for Solving Constrained Numerical Optimization. In *2007 IEEE Congress on Evolutionary Computation (CEC'2007)*, pages 670–676, Singapore, September 2007. IEEE Press.

[13] Carlos A. Coello Coello and Nareli Cruz-Cortés. Hybridizing a genetic algorithm with an artificial immune system for global optimization. *Engineering Optimization*, 36(5):607–634, October 2004.

[14] Nareli Cruz Cortés, Daniel Trejo-Pérez, and Carlos A. Coello Coello. Handling constraints in global optimization using artificial immune system. In Christian Jacob, Marcin L. Pilat, Peter J. Bentley, and Jonathan Timmis, editors, *Artificial Immune Systems. 4th International Conference, ICARIS 2005*, pages 234–247. Springer. Lecture Notes in Computer Science Vol. 3627, Banff, Canada, August 2005.

[15] Dipankar Dasgupta and Fernando Nino. *Immunological Computation: Theory and Applications*. Auerbach Publications, Boston, MA, USA, 2008.

[16] S. Forrest, A. Perelson, L. Allen, and R. Cherukuri. Self-nonself discrimination in a computer. In *IEEE Symposium on Research in Security and Privacy*, pages 202–212. IEEE Press, May 1994.

[17] H. Chueh G. C. Luh and W. W. Liu. MOIA: Multi Objective Immune Algorithm. *Engineering Optimization*, 35(2):143–164, 2003.

[18] Simon M. Garrett. How do we evaluate artificial immune systems? *Evolutionary Computation*, 13(2):145–177, 2005.

[19] F. González and D. Dasgupta. Anomaly detection using real-valued negative selection. *Genetic Programming and Evolvable Machines*, 4(4):383–403, 2003.

[20] S. Hofmeyr and S. Forrest. Architecture for the artificial immune system. *Evolutionary Computation*, 8(4):443–473, 2000.

[21] J. E. Hunt and D. E. Cooke. An adaptative, distributed learning system based on the immune system. In *Proceedings of the IEEE International Conference on System, Man and Cybernetics*, pages 2494–2499. IEEE Press, 1995.

[22] J. E. Hunt and D. E. Cooke. Recognising promoter sequences using immune algorithms. In *Proceedings of the 3rd IEEE International Conference on Intelligent Systems for Molecular Biology (ISMB)*, pages 89–97. IEEE Press, 1995.

[23] J. E. Hunt and A. Fellows. Introducing an immune reponse into a CBR system for data mining. In *Research and Development in Expert Systems XIII*, pages 34–42, 1996.

[24] Y. Ishida. An immune network model and its applications to process diagnosis. *Systems and Computers in Japan*, 24(6):646–651, 1993.

[25] A. Ishiguru, Y. Watanabe, and Y. Uchikawa. Fault diagnosis of plant system using immune network. In *Proceedings of the 1994 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI'94)*, pages 34–42, Las Vegas, Nevada, USA, October 2-5 1994. IEEE Press.

[26] N. K. Jerne. The immune system. *Scientific American*, 229(1):52–60, 1973.

[27] Kang Seok Lee and Zong Woo Geem. A new structural optimization method based on the harmony search algorithm. *Computers & Structures*, 82(9-10):781–798, April 2004.

[28] J. J. Liang, T. P. Runarsson, E. Mezura-Montes, M. Clerc, P. N. Suganthan, C. A. Coello Coello, and K. Deb. Problem Definitions and Evaluation Criteria for the CEC 2006. Technical report, Nanyang Technological University, Singapore, 2006.

[29] G. C. Luh and H. Chueh. Multi-objective optimal design of truss structure with immune algorithm. *Computers and Structures*, 82:829–844, 2004.

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

[31] Zbigniew Michalewicz and Marc Schoenauer. Evolutionary Algorithms for Constrained Parameter Optimization Problems. *Evolutionary Computation*, 4(1):1–32, 1996.

[32] L. Nunes de Castro and F.J. Von Zuben. Learning and optimization using the clonal selection principle. *IEEE Transactions on Evolutionary Computation*, 6(3):239–251, 2002.

[33] Leandro Nunes de Castro and Jonathan Timmis. aiNET: An artificial immune network for data analysis. In *Data Mining: a Heuristic Approach*, pages 231–259. Idea Group Publishing, USA, 2001.

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

[35] Ruben E. Perez and Kamran Behdinan. Particle Swarm Optimization in Structural Design. In Felix T.S. Chan and Manoj Kumar Tiwari, editors, *Swarm Intelligence: Focus on Ant and Particle Swarm Optimization*, pages 373–394. Itech Education and Publishing, Vienna, Austria, 2007. ISBN 978-3-902613-09-7.

[36] Thomas P. Runarsson and Xin Yao. Stochastic Ranking for Constrained Evolutionary Optimization. *IEEE Transactions on Evolutionary Computation*, 4(3):284–294, September 2000.

[37] 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.

[38] M. Sunar and A. D. Belegundu. Trust region methods for structural optimization using exact second order sensitivity. *International Journal for Numerical Methods in Engineering*, 32(2):275–293, 1991.

[39] J. Timmis, M. Neal, and J. Hunt. An artificial immune system for data analysis. *Biosystems*, 55(1–3):143–150, 2000.

[40] Jennifer A. White and Simon M. Garrett. Improved pattern recognition with artificial clonal selection. In Jon Timmis, Peter Bentley, and Emma Hart, editors, *Artificial Immune Systems, Second International Conference, ICARIS 2003)*, pages 181–193, Edinburgh, UK, September 2003. Springer. Lecture Notes in Computer Science Vol. 2787.

[41] J. Yoo and P. Hajela. Enhanced GA Based Search Through Immune System Modeling. In *3rd World Congress on Structural and Multidisciplinary Optimization*, Niagara Falls, New York, May 1999.

[42] J. Yoo and P. Hajela. Immune network modelling in design optimization. In David Corne, Marco Dorigo, and Fred Glover, editors, *New Ideas in Optimization*, pages 167–183. McGraw-Hill, London, 1999.