A Novel Model of Artificial Immune System for Solving Constrained Optimization Problems with Dynamic Tolerance Factor

Victoria S. Aragón, Susana C. Esquivel[1], and Carlos A. Coello Coello[2]

[1] 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
[2] CINVESTAV-IPN (Evolutionary Computation Group)**
Departamento de Computación
Av. IPN No. 2508, Col. San Pedro Zacatenco
México D.F. 07300, México
ccoello@cs.cinvestav.mx

Abstract. In this paper, we present a novel model of an artificial immune system (AIS), based on the process that suffers the T-Cell. The proposed model is used for solving constrained (numerical) optimization problems. The model operates on three populations: Virgins, Effectors and Memory. Each of them has a different role. Also, the model dynamically adapts the tolerance factor in order to improve the exploration capabilities of the algorithm. We also develop a new mutation operator which incorporates knowledge of the problem. We validate our proposed approach with a set of test functions taken from the specialized literature and we compare our results with respect to Stochastic Ranking (which is an approach representative of the state-of-the-art in the area) and with respect to an AIS previously proposed.

1 Introduction

In many real-world problems, the decision variables are subject to a set of constraints, and the search has to be bounded accordingly. Constrained optimization problems are very common, for example, in engineering applications, and therefore the importance of being able to deal with them efficiently.

Many bio-inspired algorithms (particularly evolutionary algorithms) have been very successful in the solution of a wide variety of optimization problems [1]. But, when they are used to solve constrained optimization problems, they need a special method to incorporate the problem's constraints into their fitness

* LIDIC is financed by Universidad Nacional de San Luis and ANPCyT (Agencia Nacional para promover la Ciencia y Tecnología).
** The third author acknowledges support from CONACyT project no. 45683-Y.

function. Evolutionary algorithms (EAs) often use exterior penalty functions in order to do this [2]. However, penalty functions require the definition of accurate penalty factors and performance is highly dependent on them.

The main motivation of the work presented in this paper is to explore the capabilities of a new AIS model in the context of constrained global optimization. The proposed model is based on the process that suffers the T-Cell. We also propose a dynamic tolerance factor and several mutation operators that allow us to deal with different types of constraints.

2 Statement of the Problem

We are interested in solving the general nonlinear programming problem which is defined as follows:
Find $\boldsymbol{x} = (x_1, \ldots, x_n)$ which optimizes $f(x_1, \ldots, x_n)$ subject to:

$$\begin{aligned} h_i(x_1, \ldots, x_n) = 0 \; i = 1, \ldots, l \\ g_j(x_1, \ldots, x_n) \leq 0 \; j = 1, \ldots, p \end{aligned}$$

where $(x_1, \ldots, x_n)$ is the vector of solutions (or decision variables), l is the number of equality constraints and p is the number of inequality constraints (in both cases, constraints could be linear or nonlinear).

3 Previous Work

According to [3] the main models of Artificial Immune System are: Negative Selection, Clonal Selection and Immune Network Models. They are briefly described next.

Forrest et al. [4] proposed the Negative Selection model for detection of changes. 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 of recognizing 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.

The Immune Network Model was proposed by Jerne [5], and it is a mathematical model of the immune system. In this case, the dynamics of the lymphocytes are simulated by differential equations. This model assumes that lymphocytes are an interconnected network. Several models have been derived from it [6,7].

Clonal Selection is based on the way in which both B-cells and T-cells adapt in order to match and kill the foreign cells [3]. 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 proposed by Nunes de Castro and Von Zuben [8] was originally used to solve pattern recognition and multimodal optimization problems, and there are a few extensions of this algorithm for constrained optimization. CLONALG works in the following way: first, it creates a random population of antibodies, it sorts it according to some fitness function, it clones them, it mutates each clone, it selects the fittest antibody and clones it and replaces the worst antibodies for antibodies that are randomly generated.

Those models have been used in several types of problems, but particularly, the use of artificial immune systems to solve constrained (numerical) optimization problems is scarce. The only previous related work that we found in the specialized literature is the following:

Hajela and Yoo [1] have 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 best solutions), and the other by the antibodies (which are the worst solutions). 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.

Kelsey and Timmis [9] proposed an immune inspired algorithm based on the clonal selection theory to solve multimodal optimization problems. Its highlight is the mutation operator called *Somatic Contiguous Hypermutation*, where mutation is applied on a subset of contiguous bits. The length and beginning of this subset is determined randomly.

Coello Coello and Cruz-Cortés [10] have 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.

Luh and Chueh [11] have 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 the 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.

Coello Coello and Cruz-Cortés [12] have proposed an algorithm based on the clonal selection theory for solving constrained optimization problems. The authors experimented with both binary and real-value representation, considering Gaussian-distributed and Cauchy-distributed mutations. Furthermore, they proposed a controlled and uniform mutation operator.

4 Our Proposed Model

This paper presents a novel bio-inspired model based on the T-Cells; appropriately, it is called "T-Cell Model". In a very simple way, the processes that suffer the T-Cells are the following: first, they are divided in three groups (Virgin Cells, Effector Cells and Memory Cells). Then, the natural immune system generates a huge number of virgin cells. 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. These react (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.

Thus, this model operates on three populations, corresponding to the three groups in which the T-cells are divided: (1) Virgin Cells (VC), (2) Effector Cells (EC) 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's cells into EC and EC's cells into MC. VC and EC represent their cells with binary strings using Gray coding and MC does the same, but adopting vectors of real numbers. The general structure of this model is the following:

```
Repeat a predetermined number of times
  1. Generate (in a random way) Virgin Cells
  2. Insert a percentage of Virgin Cells in Effector Cells
  3. Repeat a predetermined number of times
     3.1. Make the Effector Cells React
  End repeat
  4. Insert a percentage of Effectors Cells in Memory Cells
  5. Repeat a predetermined number of times
     5.1. Make the Memory Cells React
  End repeat
End repeat
```

4.1 Handling Constraints

In our proposed model, 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 constraints violation (sum_res)[1] and 3) the value of the objective function (only if the cell is feasible).

When the search process is driven by the value of each constraint function and the sum of constraints violation, then the selection mechanism favors feasible solutions over the infeasible ones. In this case, it is probable that, in some functions, the search falls into a local optimum. For this reason, we developed a dynamic tolerance factor (DTF), which changes with each new population, since it depends on the value of sum_res specific of the cells of the population considered (VC or EC). The DTF is calculated by adding the value of each constraint violated in each cell from a particular population (VC or EC). Then, this

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

value is divided by the number of Virgin Cells (for DTF's VC) or three times the number of Effector Cells (for DTF's EC).

When we evaluate the population using the DTF, it will be easier to generate solutions that are considered "feasible" (although they may be really infeasible if evaluated with the actual precision required). DTF relaxes the tolerance factor in order to adapt it to the particular cell into a population. This allows the exploration of each solution's neighborhood, which otherwise, would not be possible. This DTF is used by both VC and EC. In contrast, MC adopts a traditional tolerance factor, which is set to 0.0001.

4.2 Incorporating Domain Knowledge

In order to explore the frontier between the feasible and the infeasible region, EC is divided in EC_f and EC_inf. The first is composed of feasible solutions and the other of infeasible solutions. 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 if the cell is infeasible or not, respectively).

4.3 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 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 0.05. Otherwise, we change each bit from one decision variable involved in c, randomly selected, with probability 0.05.

The mutation operator for EC_f generates two mutated cells, and the best of them passes to the following iteration. This operator works in the following way:

First operator: it identifies the constraint with the most negative value (let's keep in mind that this population only has feasible cells), and changes each bit from each decision variable involved in that constraint, with probability 0.05. This operator tries to reduce the distance between the cell and the frontier with the infeasible region.

Second operator: it changes each bit from all the decision variables, with probability 0.05. If after applying mutation, a cell becomes feasible, it is inserted in EC_f according to an elitist selection. Otherwise, if after applying mutation, a cell becomes infeasible, it is inserted in EC_inf according to an elitist selection.

The mutation operator for MC applies the following equation:

$$x' = x \pm \left(\frac{N(0,1)lu - ll}{1000000gen|const||dv|} \right)^{N(0,2)} \tag{1}$$

where x and $x^{'}$ are the original and mutated decision variables, respectively. $N(0,1)$ and $N(0,2)$ refer to a randomly generated number, produced with a uniform distribution between (0,1) and (0,2), respectively. lu and ll 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 gen is the current generation number.

4.4 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, we always consider a feasible cell as better than an infeasible one. Note that before a cell is inserted into another population, it is first evaluated with the tolerance factor of the receptor population.

Therefore, the general structure of our proposed model for constrained problems is the following:

```
Repeat a predetermined number of times
  1. Randomly generate Virgin Cells
  2. Calculate DTF's VC
  3. Evaluate VC with its own DTF
  4. Insert a percentage of Virgin Cells into the Effector Cells population
  5. Repeat a predetermined number of times
    5.1. Make the Effector Cells React
    5.2. Calculate DTF's EC's
    5.3. Evaluate ECs with its own DTF
  End Repeat
  6. Insert a percentage of Effector Cells into the Memory Cells population
  7. Repeat a predetermined number of times
    7.1. Make the Memory Cells React
    7.2. Evaluate MC
  End Repeat
End Repeat
```

The most relevant aspects of our proposed model are the following:

- The fitness of a cell is determined by the value of the objective function and the value for the constrained functions.
- The size of each population is fixed. But, at first, EC_f, EC_inf and MC are empty. Step 4 is the responsible for fill EC_fand EC_inf. If at the beginning EC_f can not be filled with feasible cells from VC, the size of EC_f must to be less than the fixed value for EC_f, the following applications of step 4 could be completed it. This situation occurs for EC_inf too, but considering infeasible cells from VC. Step 6 is in charge to complete MC. First are considered the cells from EC_f and then the cells from EC_inf, if it is necessary.

- The model returns the best and worst feasible solutions in MC and the mean of the best feasible solution found in each run.
- All the equality constraints are transformed into inequality constraints, using a tolerance factor δ: $|h(\boldsymbol{x})| - \delta \leq 0$.
- VC's cells and MC's cell are sorted using the following criterion: the feasible cell 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's cells are sorted in ascending order based on their objective function values.
- EC_inf's cells are sorted in ascending order based on their sum of constraint violation.

4.5 Differences Among the Models

After the explanation of our proposed model, we have described the main differences between T-Cell and the models in Section 3. Those models are based on different immunological theories. Clonal Selection is based on the replication of antibodies according to their affinity. The Immune Network Model is a probabilistic approach to idiotypic networks. Negative Selection is based on the principles of self and nonself discrimination that takes place in the immune system. Aditionally, Negative Selection and the T-Cell Model are both based on the mechanisms of the T-Cell. However, these models give a completely different treatment to the cells (in the T-Cell Model) and the detectors (in the Negative Selection model). The Negative Selection model tries to detect a change, whereas the T-Cell Model categorizes the T-cells and it uses their phases in order to achieve different goals.

5 Experimental Setup

In order to validate our proposed model, we tested it with a benchmark of 19 test functions taken from the specialized literature [13]. 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.

Our results are compared with respect to Stochastic Ranking [14], which is a constraint handling technique representative of the state-of-the-art in the area. Additionally, we also compared our results with respect to the AIS approach reported in [12]. 25 independents runs were performed for each test problem, each consisting of 350,000 fitness function evaluations. We used a population size, for EC and MC, of 20 cells. And for VC we used 100 cells for all the test functions, except for g03 and g11, in which we used only 10 cells. We adopted a 100% and 50% replacement policy for the cells in EC and MC, respectively. All the statistical measures reported are taken only with respect to the runs in which a feasible solution was reached at the end.

6 Discussion of Results

Tables 1, 2 and 3 show the results obtained with the AIS proposed in [12], Stochastic Ranking and our T-Cell Model, respectively.

From Table 3, we can see that our model was able to reach the global optimum in 8 test functions (g01, g04, g06, g08, g11, g12, g16 and g18). Additionally, our model reached feasible solutions close to the global optimum in 7 more test functions (g02, g03, g05, g09, g10, g13 and g15) and it found acceptable (but not too close to the global optimum) feasible solutions for the rest of the test functions.

Comparing our T-Cell Model with respect to Stochastic Ranking (see Tables 2 and 3), our T-Cell Model obtained better results in 9 test functions (g02, g03, g04, g06, g10, g11, g14, g16 and g18). Both approaches found similar solutions for g01, g08 and g12. Our proposed model was outperformed in 5 test functions (g05, g07, g09, g13 and g15). With respect to the mean and worst found solutions, our model was outperformed in all functions, except for g02, g04, g06, g16 and g18.

Comparing our T-Cell Model with respect to the AIS proposed in [12] (see Tables 1 and 3), our T-Cell Model obtained better results in 9 test functions (g01, g02, g03, g05, g06, g07, g10, g11 and g13). However, for g05, our model only converged to a feasible solution in 68% of the runs while the AIS from [12] converged to a feasible solution in 90% of the runs. Both approaches found similar solutions for g04, g08 and g12. Finally, our proposed model was outperformed in g09. With respect to the mean and worst found solutions, our model was outperformed only in g07, g09 and g11.

We also conducted an analysis of variance (ANOVA) of the results obtained by our T-Cell Model and of the results obtained by Stochastic Ranking [15]. This analysis indicated that the means between the results of the algorithms had

Table 1. Results obtained by the AIS proposed in [12]. The asterisk (*) indicates a case in which only 90% of the runs converged to a feasible solution.

Function	Optimum	*Best*	*Mean*	*Worst*	*Std.Dev*
g01	-15	-14.9874	-14.7264	-12.9171	0.6070
g02	-0.803619	-0.8017	-0.7434	-0.6268	0.0414
g03	-1.0005	-1.0	-1.0	-1.0	0.0000
g04	-30665.5386	-30665.5387	-30665.5386	-30665.5386	0.0000
g05*	5126.4967	5126.9990	5436.1278	6111.1714	300.8854
g06	-6961.81387	-6961.8105	-6961.8065	-6961.7981	0.0027
g07	24.306	24.5059	25.4167	26.4223	0.4637
g08	-0.095825	-0.095825	-0.095825	-0.095825	0.0000
g09	680.63	680.6309	680.6521	680.6965	0.0176
g10	7049.33	7127.9502	8453.7902	12155.1358	1231.3762
g11	0.799	0.75	0.75	0.75	0.0000
g12	-1.0	-1.0	-1.0	-1.0	0.0000
g13	0.05395	0.05466	0.45782	1.49449	0.3790

Table 2. Results obtained by Stochastic Ranking [15]

Function	Optimum	*Best*	*Mean*	*Worst*
g01	-15	-15.0	-15.0	-15.0
g02	-0.803619	-0.803	-0.784	-0.734
g03	-1.0005	-1.0	-1.0	-1.0
g04	-30665.539	-30665.539	-30665.480	-30664.216
g05	5126.4967	5126.497	5130.752	5153.757
g06	-6961.81387	-6961.814	-6863.645	-6267.787
g07	24.306	24.310	24.417	24.830
g08	-0.095825	-0.095825	-0.095825	-0.095825
g09	680.63	680.63	680.646	680.697
g10	7049.33	7050.194	7423.434	8867.844
g11	0.799	0.750	0.750	0.751
g12	-1.0	-1.0	-1.0	-1.0
g13	0.05395	0.053	0.061	0.128
g14	-47.7648	-41.551	-41.551	-40.125
g15	961.71502	961.715	961.731	962.008
g16	-1.905155	-1.905	-1.703	-1.587
g17	8853.539	8811.692	8805.99	8559.613
g18	-0.86602	-0.866	-0.786	-0.457
g19	32.655	33.147	34.337	37.477

Table 3. Results obtained by our proposed T-Cell Model. The single asterisk (*) and double asterisk (**) indicate cases in which only 68% and 92% of the runs converged to a feasible solution, respectively.

Function	Optimum	*Best*	*Worst*	*Mean*	*Std.Dev*
g01	-15.0	-15.0	-15.0	-15.0	0.0
g02	-0.803619	-0.803102	-0.752690	-0.783593	0.013761
g03	-1.0005	-1.00041	-0.984513	-0.998627	0.004208
g04	-30665.5386	-30665.5386	-30665.5386	-30665.5386	0.0
g05*	5126.4967	5126.4982	5572.0024	5231.7186	143.0598
g06	-6961.81387	-6961.81387	-6961.81387	-6961.81387	0.0
g07	24.3062	24.3503	28.8553	25.3877	1.2839
g08	-0.095825	-0.095825	-0.095825	-0.095825	0.0
g09	680.63	680.63701	680.94299	680.74652	0.078017
g10	7049.24	7086.7891	9592.7752	7955.0428	766.493969
g11	0.7499	0.7499	0.7983	0.7553	0.010717
g12	-1.0	-1.0	-1.0	-1.0	0.0
g13	0.05394	0.054448	0.94019	0.2232	0.25325
g14	-47.7648	-44.7974	-35.6762	-41.0041	2.328270
g15	961.71502	961.72159	972.126254	964.405444	2.575551
g16	-1.905155	-1.905155	-1.905150	-1.905155	0.000001
g17 **	8853.539	8878.387	9206.116	8981.072	97.022811
g18	-0.86602	-0.86602	-0.665288	-0.811528	0.079574
g19	32.655	37.74956696	50.439198	43.997730	3.714058

significant differences except for g01, g03 and g12. The details of the analysis were, however, omitted, due to space restrictions.

We argue that our proposed model is capable of performing an efficient local search over each cell, which allows the model to improve the feasible solutions found. In cases in which no feasible solutions are found in the initial population, the mutation operators applied are capable of reaching the feasible region even when dealing with very small feasible regions.

Although there is clearly room for improvements to our proposed model, we have empirically shown that this approach is able of dealing with a variety of constrained optimization problems (i.e., with both linear and nonlinear constraints, and with both equality and inequality constraints). The benchmark adopted includes test functions with both small and large feasible regions, as well as a disjoint feasible region.

7 Conclusions and Future Work

This paper has presented a new AIS model for solving constrained optimization problems in which novel mutation operators are adopted. One of the operators incorporates knowledge of the problem, by modifying the decision variables involved in the most violated constraint. In order to get close to the frontier between the feasible and infeasible regions, it modifies the decision variables involved in the constraint farthest from zero. For some problems, the feasible region is very small, which makes it difficult to find good solutions. For this reason, we were motivated to develop a dynamic tolerance factor. Such a tolerance factor allows to explore regions of the search space that, otherwise, would be unreachable.

Our proposed model was found to be competitive in a well-known benchmark commonly adopted in the specialized literature on constrained evolutionary optimization. The approach was also found to be robust and able to converge to feasible solutions in most cases. Our analysis of the benchmark adopted made us realize that these test functions require small step sizes. Obviously, a lot of work remains to be done in order to improve the quality of the solutions found, so that the approach can be competitive with respect to the algorithms representative of the state-of-the-art in the area. For example, we plan to improve the mutation operators in order to find more quickly the frontier between the feasible and infeasible regions.

References

1. Yoo, J., Hajela, P.: Immune network modelling in design optimization. In: Corne, D., Dorigo, M., Glover, F. (eds.) New Ideas in Optimization, pp. 167–183. McGraw-Hill, London (1999)
2. Smith, A.E., Coit, D.W.: Constraint Handling Techniques—Penalty Functions. In: Bäck, T., Fogel, D.B., Michalewicz, Z. (eds.) Handbook of Evolutionary Computation, Oxford University Press and Institute of Physics Publishing (1997)

3. Garrett, S.M.: How do we evaluate artificial immune systems? Evolutionary Computation 13, 145–177 (2005)
4. Forrest, S., Perelson, A., Allen, L., Cherukuri, R.: Self-nonself discrimination in a computer. In: IEEE Symposium on Research in Security and Privacy, pp. 202–212. IEEE Computer Society Press, Los Alamitos (1994)
5. Jerne, N.K.: The immune system. Scientific American 229, 52–60 (1973)
6. Hunt, J.E., Cooke, D.E.: An adaptative, distributed learning system based on the immune system. In: Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, pp. 2494–2499. IEEE Computer Society Press, Los Alamitos (1995)
7. Ishiguru, A., Uchikawa, Y.W.: Fault diagnosis of plant system using immune network. In: MFI 1994. Proceedings of the 1994 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems, Las Vegas, Nevada, USA (1994)
8. de Castro, L.N., Von Zuben, F.: Learning and optimization using the clonal selection principle. IEEE Transactions on Evolutionary Computation 6, 239–251 (2002)
9. Kelsey, J., Timmis, J.: Immune inspired somatic contiguous hypermutation for function optimisation. In: Cantú-Paz, E., Foster, J.A., Deb, K., Davis, L., Roy, R., O'Reilly, U.M., Beyer, H.G., Kendall, G., Wilson, S.W., Harman, M., Wegener, J., Dasgupta, D., Potter, M.A., Schultz, A., Dowsland, K.A., Jonoska, N., Miller, J., Standish, R.K. (eds.) GECCO 2003. LNCS, pp. 207–218. Springer, Heidelberg (2003)
10. Coello Coello, C.A., Cruz-Cortés, N.: Hybridizing a genetic algorithm with an artificial immune system for global optimization. Engineering Optimization 36, 607–634 (2004)
11. Luh, G.C., Chueh, H.: Multi-objective optimal design of truss structure with immune algorithm. Computers and Structures 82, 829–844 (2004)
12. Cruz Cortés, N., Trejo-Pérez, D., Coello Coello, C.A.: Handling constrained in global optimization using artificial immune system. In: Jacob, C., Pilat, M.L., Bentley, P.J., Timmis, J.I. (eds.) ICARIS 2005. LNCS, vol. 3627, pp. 234–247. Springer, Heidelberg (2005)
13. Liang, J., Runarsson, T., Mezura-Montes, E., Clerc, M., Suganthan, P., Coello, C.C., Deb, K.: Problem definitions and evaluation criteria for the cec 2006 special session on constrained real-parameter optimization. Technical report, Nanyang Technological University, Singapore (2006)
14. Runarsson, T.P., Yao, X.: Stochastic Ranking for Constrained Evolutionary Optimization. IEEE Transactions on Evolutionary Computation 4, 284–294 (2000)
15. Cagnina, L., Esquivel, S., Coello, C.C.: A bi-population PSO with a shake-mechanism for solving numerical optimization. In: Proceedings of the 2007 IEEE Congress on Evolutionary Computation, Singapore, IEEE Press, Los Alamitos (2007)