Inteligencia Artificial 40(2008), 7-22

INTELIGENCIA ARTIFICIAL

http://erevista.aepia.org/

Solving Constrained Optimization using a T-Cell Artificial Immune System

Victoria S. Aragón[1], Susana C. Esquivel[1], Carlos A. Coello Coello[2]
[1]Laboratorio de Investigación y Desarrollo en Inteligencia Computacional[1]
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)
Electrical Eng. Department, Computer Science Dept.
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.

Keywords: Artificial Immune System, Constrained Optimization Problem.

1 Introduction

In many real-world problems, the decision variables are subject to a set of constraints, and the search has to be bound accordingly. Constrained optimization problems are very common, for example, in engineering applications, and therefore it is important to be 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 [31]. But, when they are used to solve constrained optimization problems, they need a special method to incorporate the problem's constraints into their fitness function. Evolutionary algorithms (EAs) often use exterior penalty functions in order to do this [27]. However, penalty functions require the definition of accurate penalty factors and performance is highly dependent on them.

Recently, several researchers have proposed novel constraint-handling techniques for EAs [3, 20, 25]. These approaches have been able to outperform penalty functions and can handle all types of constraints (linear, nonlinear, equality, inequality).

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

ISSN: 1137-3601(print); 1988-3064(on-line)

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. The remainder of the paper is organized as follows. In Section 2, we define the problem we want to solve. Section 3 describes some previous work. In Section 4, we introduce and describe our proposed model. In Section 5, we present our experimental setup. In Section 6, our results are presented and they are discussed. Finally, in Section 7, we present our conclusions and some possible paths for future work.

2 Statement of the Problem

We are interested in solving the general nonlinear programming problem which is defined as follows:

Find $\vec{x} = (x_1, \ldots, x_n)$ which optimizes $(x_1, \ldots, x_n)$ subject to:

$$h_i(x_1, \ldots, x_n) = 0 \; i = 1, \ldots, l$$
$$g_j(x_1, \ldots, x_n) \leq 0 \; j = 1, \ldots, p$$

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

Forrest et al. [26] 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 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 [11] include those reported in [26, 12, 7], among others.

The Immune Network Model was proposed by Jerne [18], 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 [13, 1]. Typical applications are [11]: detection of gene promoter sequences [14], data mining [15], diagnosis [16] and cluster analysis [17, 28].

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 [11]. 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 [23, 24] 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 antibodies and clones it and replaces the worst antibodies for antibodies that are randomly generated. Typical applications are described in [8, 24, 29], among others.

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 [30, 31] 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. This approach was tested with some structural optimization problems.

Kelsey and Timmis [19] 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 [5] 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. Additionally, the authors proposed a parallel version of the algorithm and validated it using some standard test functions reported in the specialized literature.

Balicki [2] 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, Balicki introduces a ranking procedure. This approach was validated using a constrained three-objective optimization problem.

Luh and Chueh [10, 22] 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. Other features of the approach were based on the clonal selection theory and other immunological mechanisms. CMOIA was evaluated using six test functions and two structural optimization problems.

Coello Coello and Cruz-Cortés [6] 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. This approach was tested with a set of 13 test functions taken from the specialized literature on evolutionary constrained optimization.

4 Our Proposed Model

This paper presents a novel bio-inspired model based on the T-Cell, it is called T-Cell Model. In a very simple way, the processes that suffer the T-Cell are the following: first, they are divided in three groups (Virgin Cell, 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, virgin cells becomes effector cells. These react (it means that 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 specified 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. VC and EC represent their cells with binary string using Gray coding, MC does the same, but adopting vectors of real values. 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) value of each constraint function, 2) sum of violation constraints (sum_res), it is a positive value determined by the addition of $g_i(x)^+$ for $i = 1, \ldots, p$ and $|h_k(x)|$ for $k = 1, \ldots, l$ and 3) value of objective function (only if the cell is feasible).

When the search process is driven by the value of each constraint and the sum of constraint violations, then the selection mechanisms favors the 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 develop a dynamic tolerance factor (DTF). It changes with each new population, since it depends on the value of sum_res. The DTF is calculated by adding the value of each constraint violated in each cell from a particular population (VC or EC). Then, this 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). Thus, the DTF for VC is more flexible than DTF for EC allowing that more infeasible cells being feasible cells, in a virtual way.

When we evalue 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). This allows the exploration of each solution's neighborhood, which otherwise, would not be possible. This DTF is used by both VC and EC. If the value of DTF is lower than 0.0001, we set it to 0.1 and 0.001 for VC and EC, respectively. In contrast, MC adopts a traditional tolerance factor, which is set to 0.0001. The cells within MC need to be evaluated with the traditional tolerance factor because these are the real solution for the problem.

4.2 Incorporating Domain Knowledge

In order to explore the frontier between feasible and infeasible zones, EC is divided in EC_f and EC_inf. The first is composed by feasible solutions and the other by infeasible solutions. Also, we introduce domain knowledge through the mutation operator, which modify the decision variables involve in the constraint with the highest violation.

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 the total number of constraints, then we change each bit from each decision variable involve in c with a random probability between 0.01 and 0.2. Otherwise, we change each bit from one decision variable involve in c, randomly selected, with a random probability between 0.01 and 0.2. We use a random probability because after some experiments, we observed that some test functions required different step sizes. If after applying mutation, a cell becomes feasible, it is inserted in EC_f according to an elitist selection.

The mutation operator for EC_f works in the following way: it changes each bit from all decision variables, with a random probability between 0.001 and 0.2. This random probability has the same motivation that the previously.

The mutation operator for MC applies the following equation:

$$x^{'} = x \pm \left(\frac{N(0,1)lu - ll}{10^m gen|const||dv|} \right)^{N(0,1)} \quad (1)$$

where x and $x^{'}$ are the original and mutated decision variables, respectively. $N(0,1)$ refers to a random number with a uniform distribution between (0,1). lu and ll are the upper and lower limits of x. $|const|$ refers to the number of constraints. $|dv|$ refers to the number of decision variables of the problem, gen is the current generation number and m is an integer (its value is seted in Section 5).

4.4 Replace Mechanisms

The replace mechanisms are always applied in an elitist way, both within a population and between different populations. They take into account the value of objective function or the sum of constraint 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 Effector Cells population
5. Calculate DTF's EC's
6. Repeat 50 times
   6.1. Make the Effector Cells React
   6.2. Evaluate EC's with its own DTF
   End repeat.
7. Insert a percentage of Effectors Cells into Memory Cells population
8. Repeat 100 times
   8.1. Make the Memory Cells React
   8.2. Evaluate MC
   End repeat.

End repeat.

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

- All equality constraints are converted into inequality constraints, $|h(\vec{x})| - \delta \leq 0$, using a tolerance factor.
- VC's cells and MC's cell are sorted using the following criterion: the feasible cell whose objective function 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 on their objective function.
- EC_inf's cells are sorted in ascending order on their sum of constraint violation.

4.5 Differences between the Models

The immune system models described in 3 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-nonself discrimination that take place in the immune system. Aditionally, Negative Selection and 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 T-Cell Model) and detectors (in Negative Selection). The Negative Selection Model tries to detect some change, whereas T-Cell Models categorizes the T-cell and it uses their phases in order to achieve different goals.

5 Esperimental Setup

In order to validate our proposed model we tested it with a benchmark of 19 test functions taken from the specialized literature [4]. The functions g02, g03, g08 and g12 are maximization problems (for simplicity, these problems were converted into minimization problems using $-f(x)$) and the rest are minimization problems.

Our results are compared with respect to Stochastic Ranking, we take its result from [21], which is a constraint handling technique representative of the state-of-the-art in the area, and with respect to the AIS approach reported in [6]. For equation 1, we used $m = 10^7$ for all functions except for g02, here we used $m = 10^2$. 25 independents runs were performed for each problem, each consisting of 350,000 fitness function evaluations. We experimented with different population sizes, the best results were obtained using: 1) for VC 100 cells for all functions, except for g19 here we used 10 cells and for g10 and g15 we used 20 cells, 2) for EC_f, EC_inf and MC we used 20 cells for all functions, except for g10 and g19, here we used 10 cells. We adopted a 100% and 50% replacement for the cells in EC's 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 [6], Stochastic Ranking and our T-Cell Model, respectively. Figures 1 to 16 show graphically the best and mean values obtained for all test functions, except for g08, g11 and g12 where all algorithms found the optimum values.

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, g15 and g16). Additionally, our model reached feasible solutions close to the global optimum in 7 more test functions (g02, g03, g07, g09, g13, g14 and g18) and it found acceptable (but not too close from the global optimum) feasible solutions for the rest of the test functions.

Comparing T-Cell Model with respect to Stochastic Ranking (see Tables 2 and 3), T-Cell Model obtained better results in 9 test functions (g03, g04, g06, g11, g14, g15, g16, g17 and g18). Both approaches found similar solutions for g01, g08 and g12. Our model was outperformed in 7 functions (g02, g05, g07, g09, g10, g13 and g19) . With respect to the mean and worst found solutions, our model was outperformed all functions except g03, g04, g06, g11, g14 and g16.

Comparing T-Cell Model with the AIS proposed in [6] (see Tables 1 and 3), T-Cell Model obtained better results in 8 test functions (g01,g02, g03, g05, g06, g07, g10 and g11). Both approaches found similar solutions for g04, g08 and g12. Finally, our model was outperformed in g09 and g13. With respect to the mean and worst found solutions, our model was outperformed only in g02, g07, g09 and g13.

We conducted an analysis of variance of the results obtained by our T-Cell Model and of the results obtained by Stochastic Ranking [21]. Due to, for some functions, the results do not follow a normal distribution, we used the Kruskal Wallis test [9] and then Turkey method [9]. The first test indicates if the means between the results of the algorithms had significant differences and the second one indicates in which experimental conditions the means had significant differences. Table 4 shows the values obtained for these tests. The first column represents to the function, the second column shows the values for Kruskal Wallis test (the means had significant differences if this value p is lower than 0.05), the third and fourth column indicate the lower and upper limits (if the values contained inside this interval does not contain the zero then the means had significant differences). After the analysis of Table 4, we observed that for all function the means have significant differences except for g11. Note that we do not apply these tests to g01, g08 and 12 because, for these functions both algorithms found the optimum solution in all runs.

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

Although there is clearly room for improving 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 objective function, 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.

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	7049.24	7127.9502	8453.7902	12155.1358	1231.3762
g11	0.7499	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 1: Results obtained with AIS proposed in [6]. The asterisk (*) indicates a case in which only 90% of the runs converged to a feasible solution

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	7049.24	7050.194	7423.434	8867.844
g11	0.7499	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 2: Results obtained with Stochastic Ranking [21]

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 involve in the most violated constraint. For some functions, the feasible

Function	Optimum	*Best*	*Worst*	*Mean*	*Std.Dev*
g01	-15.0	-15.0	-15.0	-15.0	0.0
g02	-0.803619	-0.802914	-0.301795	-0.546031	0.168392
g03	-1.0005	-1.000499	-1.000498	-1.000499	0.000001
g04	-30665.5386	-30665.5386	-30665.5386	-30665.5386	0.0
g05*	5126.4967	5126.6595	5850.9358	5307.1073	230.2466
g06	-6961.81387	-6961.81387	-6961.81387	-6961.81387	0.0
g07	24.306	24.3118	28.5089	25.8927	1.1297
g08	-0.095825	-0.095825	-0.095825	-0.095825	0.0
g09	680.63	680.6312	680.7411	680.6730	0.030547
g10	7049.24	7061.67	7894.75	7451.88	218.39739
g11	0.7499	0.7499	0.7499	0.7499	0.0
g12	-1.0	-1.0	-1.0	-1.0	0.0
g13	0.05395	0.054879	2.03033	0.64231	0.534641
g14	-47.7648	-46.2546	-40.2996	-43.6876	1.538386
g15	961.71502	961.71502	971.43611	065.02171	3.10270
g16	-1.905155	-1.905155	-1.905155	-1.905155	0.0
g17	8853.539	8862.383	9271.390	8984.399	117.5927
g18	-0.86602	-0.866019	-0.66920	-0.78805	0.09285
g19	32.655	34.649	73.151	52.617	10.1005

Table 3: Results obtained with our proposed T-Cell Model. The asterisk (*) indicates a case in which only 96% of the runs converged to a feasible solution

Function	p	lower limit	upper limit
g02	2.54392e-009	16.0252	31.7348
g03	4.53296e-011	-35.0356	-18.9644
g04	8.98673e-011	17.4421	32.5579
g05	2.17934e-009	16.2556	32.0911
g06	9.06124e-011	17.4406	32.5594
g07	2.93747e-009	15.4582	30.7018
g09	1.08889e-008	14.9037	30.4563
g10	1.74435e-008	14.5837	30.1363
g11	0.1298	-1.7753	13.8553
g13	3.35698e-010	17.1443	32.6957
g14	0.0009	-21.3391	-5.4609
g15	2.01142e-008	14.9660	31.0340
g16	3.97653e-011	-32.4185	-17.5815
g17	3.60989e-010	17.1853	32.8147
g18	7.25903e-010	16.6646	32.2157
g19	3.15542e-010	16.8670	32.1330

Table 4: Analysis of Variance

region is very small, which makes it difficult to find good solutions. For this reason, we were motivated to develop a dynamic tolerance factor. It allows to explore regions of the search space that, otherwise, would be unreachable, if we use a tolerance factor very restrictive.

The 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, except for g02, due to this function has a feasible region bigger than the other functions. A lot of work remains to be done in order to improve the quality of some 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 the frontier and feasible zone faster. Nevertheless, it is important to emphasize that there is very little work regarding the use of artificial immune systems for constrained numerical optimization, and in that context, this approach provides a viable alternative.

Acknowledgements

The first two authors acknowledge support from the Universidad Nacional de San Luis and the ANPCYT. The third author acknowledges support from the Consejo Nacional de Ciencia y Tecnología (CONACyT) through project number 42435-Y.

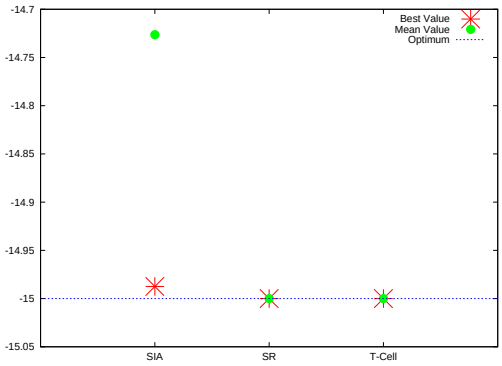

Figure 1. Best and Mean Values for g01

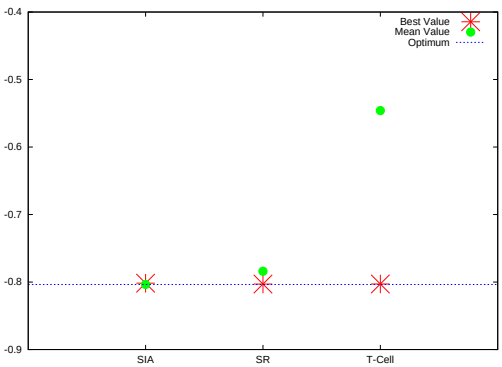

Figure 2. Best and Mean Values for g02

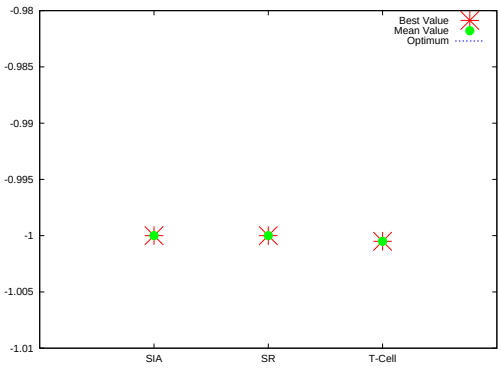

Figure 3. Best and Mean Values for g03

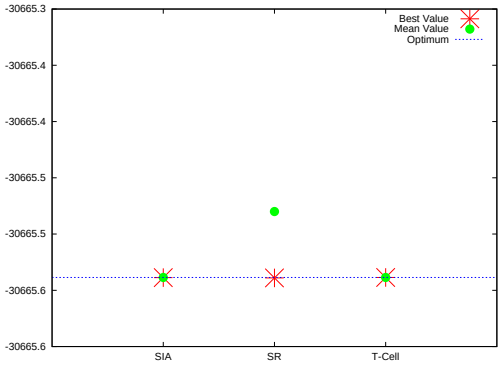

Figure 4. Best and Mean Values for g04

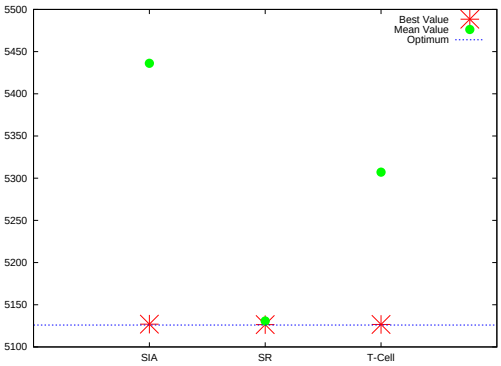

Figure 5. Best and Mean Values for g05

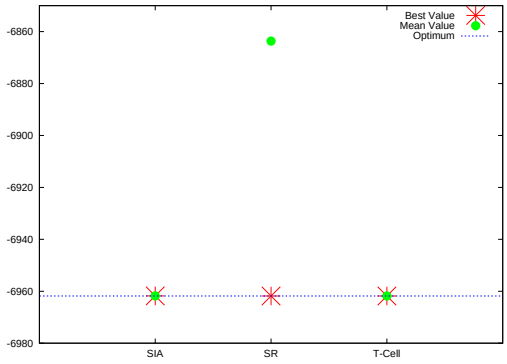

Figure 6. Best and Mean Values for g06

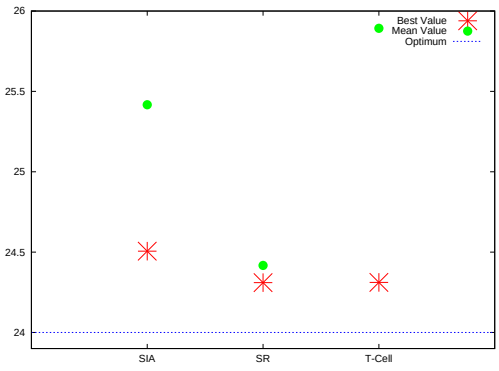

Figure 7. Best and Mean Values for g07

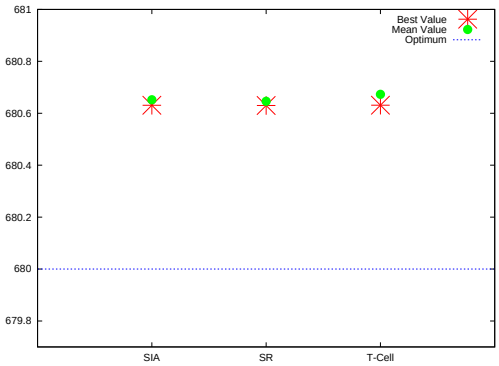

Figure 8. Best and Mean Values for g09

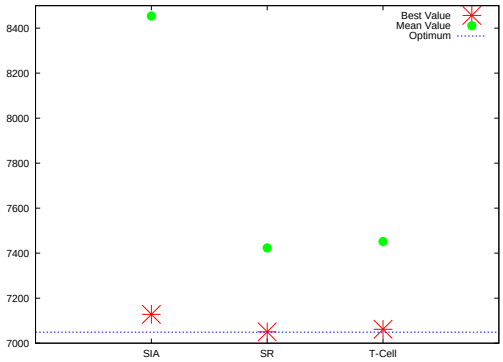

Figure 9. Best and Mean Values for g10

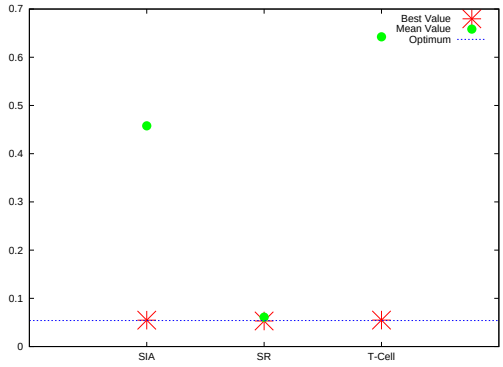

Figure 10. Best and Mean Values for g13

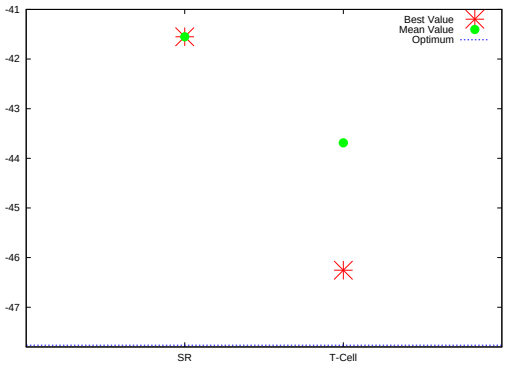

Figure 11. Best and Mean Values for g14

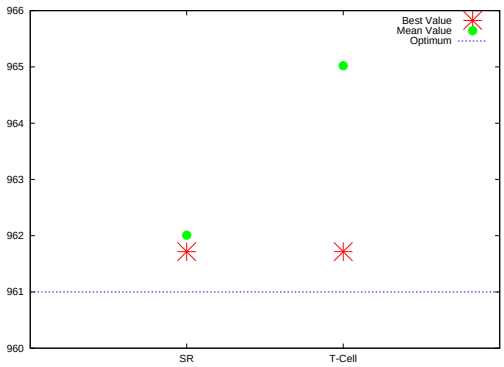

Figure 12. Best and Mean Values for g15

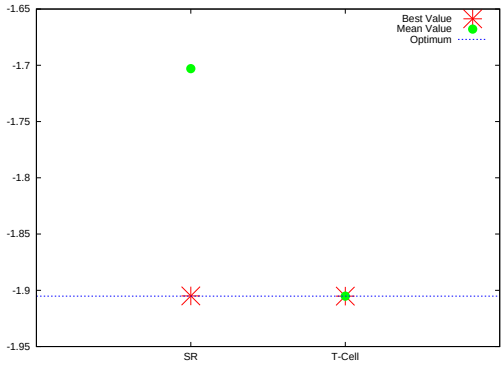

Figure 13. Best and Mean Values for g16

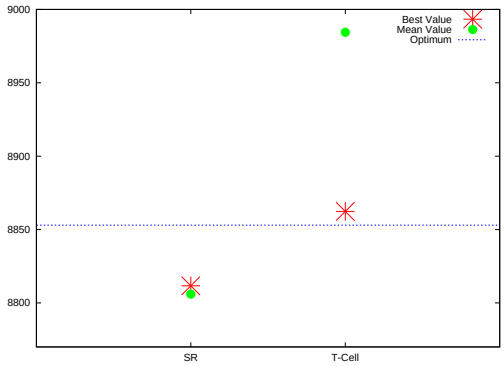

Figure 14. Best and Mean Values for g17

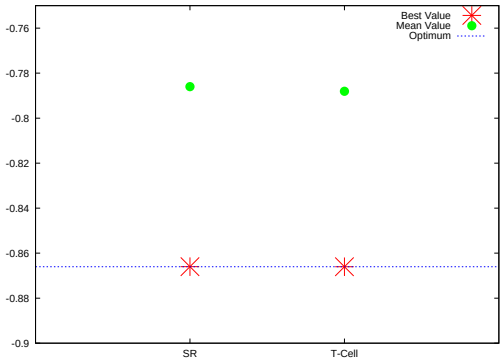

Figure 15. Best and Mean Values for g18

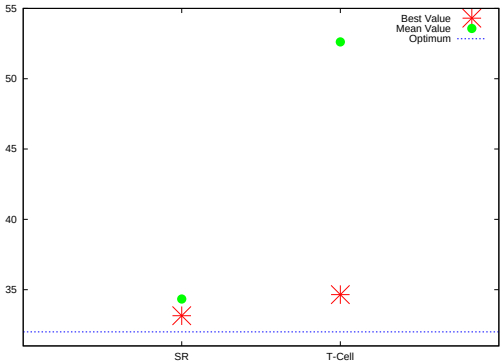

Figure 16. Best and Mean Values for g19

References

[1] Y. Watanable A. Ishiguru and Y. Uchikawa. Fault diagnosis of plant system using immune network. In Procceding of the 1994 IEEE International Conference on Multisensor Fusion and Integration for Intelligent System (MFI'94), Las Vegas, October 2-5, 1994.

[2] 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.

[3] Carlos A. Coello Coello. Theoretical and Numerical Constraint Handling Techniques used with Evolutionary Algorithms: A Survey of the State of the Art. *Computer Methods in Applied Mechanics and Engineering*, 191(11-12):1245–1287, January 2002.

[4] J. Liang T. Runarsson E. Mezura-Montes M. Clere P. Suganthan C. Coello Coello and K. Deb. Problem definitions and evaluation criteria for the cec 2006 special session on constrained real-parameter optimization. Technical Report, Nanyang Technological University, 2006.

[5] 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.

[6] Nareli Cruz Cortés, Daniel Trejo-Pérez, and Carlos A. Coello Coello. Handling constrained 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.

[7] D. Dasgupta and S. Forrest. An anomaly detection algorithm inspired by the immune system. In Artificial Immune System and Their Applications, chapter 14, pages 262-277. Springer-Verlag, Inc, 1999.

[8] L. de Castro and J. Timmis. Ainet: An artificial immune network data analysis. In Abbass, H. A., Sarker, R. A., and Newton, C. S., editors, Data Mining: A Heuristic Approach, chapter XII, pages 231-259. Idea Group Publishing, USA, 2001.

[9] Bernabe Dorronsoro. Análisis de varianza (ANOVA) con MATLAB MINI-HOWTO.

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

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

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

[13] J. E. Hunt and D. E. Cooke. An adaptative, distributed learning system based on the immune system. In Procceding of the IEEE International Conference on System, Man and Cybernatics, pages 2494-2499, 1995.

[14] J. E. Hunt and D. E. Cooke. Recognising promoter sequences using immune algorithms. In Proceeding of the 3rd IEEE International Conference on Intelligent System for Molecular Biology (ISMB), pages 89-97, 1995.

[15] J. E. Hunt and A. Fellows. Introducing an immune reponse into a cbr system for data mining. 1996.

[16] Y. Ishida. An immune network model and its applications to process diagnosis. System and Computer in Japan, 24(6):646-651, 1993.

[17] M. Neal J. Timmis and J. Hunt. An artificial immune system for data analysis. Biosystem, 55(1/3):143-150, 2000.

[18] N. K. Jerne. The imune system. Scientific American, 229(1):52-60, 1973.

[19] Johnny Kelsey and Jon Timmis. Immune inspired somatic contiguous hypermutation for function optimisation. In Erick Cantú-Paz, James A. Foster, Kalyanmoy Deb, Lawrence Davis, Rajkumar Roy, Una-May O'Reilly, Hans-Georg Beyer, Russell K. Standish, Graham Kendall, Stewart W. Wilson, Mark Harman, Joachim Wegener, Dipankar Dasgupta, Mitchell A. Potter, Alan C. Schultz, Kathryn A. Dowsland, Natasa Jonoska, and Julian F. Miller, editors, *GECCO*, volume 2723 of *Lecture Notes in Computer Science*, pages 207–218. Springer, 2003.

[20] Slawomir Koziel and Zbigniew Michalewicz. Evolutionary Algorithms, Homomorphous Mappings, and Constrained Parameter Optimization. *Evolutionary Computation*, 7(1):19–44, 1999.

[21] S. Esquivel L. Cagnina and C. Coello Coello. A bi-population pso with a shake-mechanism for solving numerical optimization. In Proceedings of the Congress on Evolutionary Computation 2007.

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

[23] L. Nunes de Castro and J. Timmis. An artificial immune network for multimodal function optimization. In *Proccedings of the 2002 Congress on Evolutionary Computation (CEC'2002)*, volume 1, pages 669–674, Honolulu, Hawaii, May 2002.

[24] 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.

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

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

[27] Alice E. Smith and David W. Coit. Constraint Handling Techniques—Penalty Functions. In Thomas Bäck, David B. Fogel, and Zbigniew Michalewicz, editors, *Handbook of Evolutionary Computation*, chapter C 5.2. Oxford University Press and Institute of Physics Publishing, 1997.

[28] J. Timmis and M. Neal. A resource limited artificial immune system for data analysis. In Procceding of ES2000, pages 19-32, Cambridge, UK, 2000.

[29] J. White and S. Garret. Improved pattern recognition with artificial clonal selection. In Procceding of the 2nd International Conference on Artificial Immune Systems (ICARIS-03), pages 181-193, 2003.

[30] 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.

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