Hybridizing an Immune Artificial Algorithm with Simulated Annealing for Solving Constrained Optimization Problems

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
[2] CINVESTAV-IPN (Evolutionary Computation Group)‡
Departamento de Computación - Av. IPN No. 2508, Col. San Pedro Zacatenco
México D.F. 07300, MÉXICO

Abstract. In this paper, we present a modified version of an algorithm inspired on the T-Cell model, it is an artificial immune system (AIS), based on the process that suffers the T-Cell. The proposed model (TCSA) is increased with simulated annealing, for solving constrained (numerical) optimization problems. We validate our proposed approach with a set of test functions taken from the specialized literature. We indirectly compare our results with respect to GENOCOP III, a well known software based on genetic algorithm.

Keywords: Artificial Immune System, Constrained Optimization Problem, Simulated Annealing.

1 Introduction

Over the last years, a bio-inspired system has call the attention of some researchers, the Natural Immune System (NIS) and its powerful capacity of information processing. The NIS is a very complex system with several defense mechanisms against foreign organisms. The main purpose of the NIS is recognize all cells of the host and categorize them in order to induce the appropriate immune response. The NIS learns through the evolution to distinguish between self and non-self. Besides, it has many desirable characteristics from the point of view computational, such as: uniqueness, pattern recognition, diversity, tolerance faults, learning and memory, self-organization, robustness, cooperation between different layers, among others. Thus, these characteristics and a well-known about the functionality of the NIS are excellent motivations to develop

† 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.
⋆

Artificial Immune Systems (AIS) to hand constrained problems. Besides, this kind of heuristic has not been frequently used for solving constrained problems.

The main motivation of the work presented in this paper is to verify the behavior of this new version of T-Cell Model [2] which includes simulated annealing, as a local search, in order to improve the best found solution and decrease the number of objective function evaluations performed by the algorithm without negatively affect the quality of the solutions, in the context of constrained problems.

The remainder of the paper is organized as follows. In Section 2, we define the problem we want to solve. Section 3 describe the artificial immune systems existing and some previous work, respectively. In Section 4 describes the proposed algorithm. In Section 5, we present our experimental setup and results, for last Section 6 our conclusions and some possible paths for future work are presented.

2 Statement of the Problem

In a general way, a minimization problem can be expressed as:

minimize

$$f(X) \qquad i = 1, \ldots, n \tag{1}$$

here f designates the objective function and $X = (x_1, x_2, \ldots, x_n)^T$ the design variables vector ($x_i^l \leq x_i \leq x_i^u$). f is subjected to some functions. They are inequality constraints ($g_j(X) \leq 0$, $j = 1, \ldots, m$), equality constraints ($h_k(X) = 0$, $k = 1, \ldots, l$) and side constraints with lower and upper limits indicated by the superscripts l and u, respectively. These functions, which can be solved analytically or numerically, may be linear or non-linear and contain the design variables in an explicit or a non-explicit form.

3 Previous Work

According to [14] the main Artificial Immune System models are: Negative Selection [13],[14], Clonal Selection [11, 18] and Immune Network Models [15] and [14].

These SIA 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 previous related work that we found in the specialized literature is described next.

Hajela and Yoo 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 [19, 20].

Coello Coello and Cruz-Cortés 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 [9].

Coello Coello and Cruz-Cortés 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 [10].

Bernardino, Barbosa and Lemonge [8] proposed a genetic algorithm hybridized with an artificial immune system (AIS-GA). The AIS is inspired in the clonal selection principle and it is embedded into a standard GA search engine in order to help move the population into the feasible region. They also present a modified version of AIS-GA, an AIS-GA with a clearing procedure AIS-GA^C. This procedure is applied over the union of the new population and the previous one, in order to create the new population. Both approaches are applied over six mechanical engineering optimization problems.

4 Proposed Algorithm Based on TCELL

Here, an adaptive immune system model based on the immune responses mediated by the T cells is adopted. Originally, this approach was used to solve static optimization problems [3] and then it was extended to solve dynamic problems, constrained problems and dynamic constrained problems [4], [5], [6], [1] and [7] . It considers many of the processes that T cells suffer from their origin in the hematopoietic stem cells in the bone marrow until they become memory cells.

TCSA (Constrained T-Cell with Simulated Annealing) is an algorithm inspired on the TCELL model [1], which we modified and add to it the well known simulated annealing technique to solve constrained optimization problems. TCSA operates on four populations, corresponding to the groups in which the T-cells are divided: (1) Virgin Cells (VC), (2) Effector Cells with cluster denomination CD4 (CD4), (3) Effector Cells with cluster denomination CD8 (CD8) and (4) Memory Cells (MC). Each population is composed by a set of T cells whose characteristics are subject to the population to which they belong.

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

coding (called TCR_b) and 2) a *TCR* represented by a vector of real numbers (called TCR_r).

In our proposed algorithm, positive selection is in charge of eliminating the cells that recognize the antigen with a low matching. On the other hand, negative selection has to eliminate the cells that have a similar TCR, according to a Hamming or an Euclidean distance, depending on whether the TCR is represented by a TCR_b or by a TCR_r.

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

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

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

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

At the beginning, the proliferation level of each stimulated cell, ce_i, is given by a random value within [1, 3],[3] but then, it is determined taking into account the proliferation level of its stimulating cell (ce_j). If the ce_i is better than ce_j, then ce_i keeps its own proliferation level; otherwise, ce_i receives a level which is 10% lower than the level of ce_j.

Memory cells proliferate and differentiate according to their proliferation level a random value within [1, 2],[4] and differentiation level (number of decision variables[5]), respectively. Both levels are independent from the other memory cells.

In TCSA algorithm, the constraint-handling method needs to calculate, for each cell (solution), regardless of the population to which it belongs, the following: 1) the sum of constraint violations (sum_res)[6] and 2) the value of the objective function (only if the cell is feasible).

We also use a dynamic tolerance factor (DTF), on equality constraints, in order to not remove all infeasible solutions from populations. DTF virtually

[3] This value was derived after numerous experiments.

[4] This value was derived after numerous experiments.

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

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

expand feasible regions increasing the traditional tolerance factor according to the sum of constraint violations of the cell. This DTF is used by CD4 and CD8 populations.

We consider a ce_i cell is better than a ce_j cell if 1) TCR's ce_i is feasible and TCR's ce_j is infeasible, 2) both cells have feasible TCRs but objective function value's ce_i is lower than objective function value's ce_j and 3) both cells have infeasible TCRs but sum_res' ce_i is lower than sum_res' ce_j. This criterion is used to perform population sort. Each type of cell has its own differentiation process, which is blind to their representation and population.

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

Differentiation for CD8: the differentiation level for cell ce_i is related to its stimulating cell (ce_j). If the TCR_r of the ce_j is better than the TCR_r of the stimulated cell ce_i , then the level (for ce_i) is a random number within [$\mid dv/2 \mid$, $\mid dv \mid$[7]; otherwise, it is a random value within [1, $\mid dv \mid /2$], where $\mid dv \mid$ is the number of decision variables of the problem. Each variable to be changed is chosen in a random way and it is modified according to $x^{'} = x \pm r$ where $r = \frac{U(0,lu-ll)}{10iter}^{U(0,1)}$, x and $x^{'}$ are the original and the mutated decision variables, respectively. lu and ll are the upper and lower bounds of x, respectively. $iter$ indicates the number of iterations until reaching the maximum number of evaluations. At the moment of the differentiation of a cell (ce_i), the value of the objective function of its stimulating cell (ce_j) is taken into account. In order to determine if r will be added or subtracted to x, the following criteria are considered: if ce_j is better than ce_i and the decision variable value of ce_j is less than the value of ce_i, or if ce_i is better than ce_j and the decision variable value of ce_i is less than the value of ce_j, then r is subtracted from x; otherwise, r is added to x. Both criteria aim to guide the search towards the best solutions found so far.

Differentiation for MC: each variable to be changed is chosen in a random way and it is modified using the following equation according to the differentiation level: $x^{'} = x \pm r$ where $r = \left(\frac{U(0,lu_x-ll_x)}{10^7 iter}\right)^{U(0,1)}$, x and $x^{'}$ are the original and the mutated decision variables, respectively. $U(0,w)$ refers to a random number with a uniform distribution in the range (0,w). lu_x and ll_x are the upper and lower bounds of x, respectively. $iter$ indicates the number of iterations until reaching the maximum number of evaluations for a change. In a random way, we decide if r will be added or subtracted to x. If after ten try the procedure can not find a x' in the allow range a random number with a uniform distribution is signed to it.

The general structure of our proposed algorithm for constrained optimization problems is given next.

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

TCSA Algorithm

```
Initialize_VC();
Evaluate_VC();
Assign_Proliferation();
Divide_CDs();//
Positive_Selection();// eliminate the worst cells
Negative_Selection();// eliminate the most similar cells
while (A number of evaluations has not been performed) do
  for i=1 to rep
    Activate_CD4();
  endfor
  Sort_CD4();
  Best_CD4_pass_CD8();
  for i=1 to rep
    Activate_CD8();
  endfor
  Sort_CD8();
  Insert_CDs_en_MC();
  for i=1 to rep
    Activate_MC();
  endfor
  Sort_CM();
od
Simulated_Annealing();
Statistics();
```

The algorithm works in the following way. At the beginning, the TCR_b and TCR_r from the virgin cells are initialized in a random way, according to the TCR's encoding. Then, each TCR of a virgin cell is evaluated. Then, the proliferation levels are assigned. The virgins cells are divided taking into account their feasibility. Next, TCR_b and TCR_r feasible from VC are selected to form CD4 and CD8, respectively. If it is not possible to complete the required number of cells for population the infeasible TCRs are selected. Each effector cell will inherit the proliferation level of the virgin cell which received the TCR.

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

A maximum number of objective function evaluations is allowed. Then the actions are: to activate the CD4 population *rep* times; in other words, to perform proliferation and differentiation of all the cells from CD4. The best cell between the original cell and its clones is passed to the next iteration. Then, these cells

are sorted. The best solution from CD4 is used to replace the worst solution in CD8. Next, the CD8 population is activated *rep* times and sorted.

The best solutions from CD4 and CD8 are inserted or are used to replace the worst solutions in MC (depending on whether or not, MC is empty). Since the representation schemes of the TCR, for CD4 and MC, are different, before the insertion of the best cell from CD4 (with TCR_b) into MC, the receptor has to be converted into a real-values vector (TCR_r). Next, the cells from MC are activated a certain (predefined) number of times, *rep*.

To the best found solution we apply it the well known technique Simulated Annealing, source code in C was taken from `http://www.taygeta.com/annealing`. We use default parameters, except for the number of iterations, we use 100 instead 400.

5 Numerical Experiments and Results

In order to validate TCSA we test it with fourteen constrained problems. Four (G) test problems from [16] and benchmark of 10 general geometric programming (GGP) test problems from [12]. Table 1 shows the main characteristics of these problems. Some preliminary runs were performed, thus, the best results were found with the following parameters. We used a population size, for VC of 200 cells. For CD4 and CD8 we used 5 cells and 2 cells for MC. The mutation probability $\text{prob}_{diff-CD4}$ was 0.5. The number of times that react the CD4, CD8 and MC populations was 10, except for GGP3, it was 100 times. 30 independents runs were performed for each test problem. Measures reported are taken only with respect to the runs in which a feasible solution was reached at the end. These measures include the percentage deviation of the average (% Aver. Error) and the best (% Best Error) objective function value from the optimum ($(f - optimum)/ \mid optimum \mid \times 100$) and the average computational effort (CEff) need to reach a solution that satisfies $\mid f - optimum \mid \leq 10^{-4} optimum + 10^{-6}$ or a maximum number of objective function evaluations be reach. Our results, are indirectly compared with respect to GENOCOP III [17] a well known algorithm. It performs 10 independent runs. The results are showed in table 2. The values in **bold** and *italic* indicate that the algorithm found the optimum and best value, respectively. For last - indicates any feasible solution was found.

All solutions, for all test problems, found by TCSA are feasible. We can see that our algorithm was able to reach solutions which errors are close to zero in five test functions (G1, G3, GGP5, GGP6 and GGP7), in both mean and best percentage error. But for G5, GGP1, GGP2, GGP3, GGP4, GGP8, GGP9 and GGP10 the only errors close to zero are for the best found solutions.

Comparing our average performance with respect to GENOCOP III our TCSA obtained better results in ten test problems (G3, G5, GGP1, GGP2, GGP4, GGP6, GGP7, GGP8, GGP9 and GGP10). TCSA was outperformed in remaining four test problems. Comparing our best performance with respect to GENOCOP III our TCSA obtained better results in ten test problems (G3, G5, GGP1, GGP2, GGP3, GGP4, GGP7, GGP8, GGP9 and GGP10). Our pro-

Table 1. Test Problems

Problem	dimension	inequality constraints	equality constraints	optimum	Maximum Evaluations
G1	13	9	0	-15.0	8,000
G2	8	6	0	7049.33	52,800
G3	7	4	0	680.6301	46,200
G5	10	8	0	24.3062	46,500
GGP1	7	14	0	1227.23	9,000
GGP2	6	1	4	-0.3888	10,000
GGP3	8	6	0	7049.24	7,970
GGP4	8	4	0	3.9511	17,900
GGP5	5	6	0	10122.6964	12,800
GGP6	3	1	0	-83.2535	16,100
GGP7	4	2	0	-5.7398	12,100
GGP8	8	4	0	-6.0482	18,000
GGP9	10	7	0	1.1437,	19,900
GGP10	11	9	0	0.1406	300,000

Table 2. GGP test instances

Problem	TCSA			GENOCOP III		
	% Aver. Error	% Best Error	CEff	% Aver. Error	% Best Error	CEff
G1	*0.02*	**0.00**	7985	**0.00**	**0.00**	8393
G2	71.30	3.38	52,024	*5.59*	*0.54*	52,460
G3	*0.03*	**0.00**	40040	0.05	0.01	46,165
G5	*5.08*	*0.60*	46,404	40.43	8.67	46,462
GGP1	*10.03*	*0.82*	9,008	10.08	4.52	9,045
GGP2	*6.54*	**0.00**	8,597	-	-	-
GGP3	38.32	*0.62*	8,105	*12.35*	2.60	7,970
GGP4	*3.36*	*0.07*	17,909	5.54	0.89	17,975
GGP5	0.21	**0.00**	12,661	*0.20*	**0.00**	12,866
GGP6	*0.10*	**0.00**	14,119	0.12	**0.00**	16,115
GGP7	*0.05*	**0.00**	9,550	0.42	0.02	12,145
GGP8	*1.32*	*0.03*	18,005	2.46	0.18	18,023
GGP9	*34.22*	**0.00**	18,631	75.18	40.22	19,990
GGP10	*28.72*	*0.01*	300,000	-	-	-

posed algorithm was outperformed in G2. Note both algorithm have the same best performance in G1, GGP5 and GGP6. Only for GGP3 TCSA performed more objective function evaluations than GENOCOP III.

GENOCOP III was not able to find feasible solutions for GGP2 and GGP10 while TCSA was it. If worth noting that TCSA uses a local search whilst GENOCOP III does not. This fact could help TCSA in those cases where GENOCOP III could not find any feasible solution. Even when for some cases the TCSA's performance is not good it can find feasible solutions in all runs.

6 Conclusions and Future Work

This paper presents a modified version of the Artificial Immune System T-Cell Model for solving constrained optimization problems which includes a simulated annealing at the end of the search process in order to improve the best found solution, it was called TCSA.

Our proposed algorithm was found to be competitive in the benchmark used. The approach was able to converge to feasible solutions in all cases tested. 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.

As future work, we plan to improve the mutation operators in order to find more quickly the frontier between the feasible and infeasible regions and also to improve the quality of the found solutions.

References

1. Victoria S. Aragón. *Optimización de Problemas con Restricciones a través de Heurísticas BioInspiradas*. PhD thesis, Universidad Nacional de San Luis, 2010.
2. 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, *Artificial Intelligence - MICAI07*, pages 19–29, Aguascalientes, México, 2007. Springer. Lecture Notes in Artificial Intelligence Vol. 4827.
3. Victoria S. Aragón, Susana C. Esquivel, and Carlos A. Coello Coello. Artificial Immune System for Solving Global Optimization Problems. In *XIV Congreso Argentino en Ciencias de la Computacion (CACIC 2008)*, pages 637–647, 2008. Chilecito, La Rioja, Argentina.
4. 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.
5. 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.

6. Victoria S. Aragón, Susana C. Esquivel, and Carlos A. Coello Coello. A modified version of a t-cell algorithm for constrained optimization problems. *International Journal for Numerical Methods in Engineering*, 84(3):351–378, 2010.
7. Victoria S. Aragón, Susana C. Esquivel, and Carlos A. Coello Coello. A t-cell algorithm for solving dynamic optimization problems. *Information Sciences*, 181(17):3614–1637, September 2011.
8. 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.
9. 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.
10. 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.
11. L. N. 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.
12. C. Floudas, P. Pardalos, C. Adjiman, W. Esposito, Z. Gümüs, S. Harding, J. Klepeis, C. Meyer, and C. Schweiger. *Handbook of Test Problems in Local and Global Optimization*. Kluwer Academic Publishers, Dordrecht, 1999.
13. 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.
14. Simon M. Garrett. How do we evaluate artificial immune systems? *Evol. Comput.*, 13(2):145–177, 2005.
15. N. K. Jerne. The immune system. *Scientific American*, 229(1):52–60, 1973.
16. Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, Berlin, 1994.
17. N. Mladenović, M . Dražic, V. Kovacevic Vujcic, and M. Cangalovic. General variable neighborhood search for the continuous optimization. *European Journal of Operational Research*, 191:753–770, 2008.
18. 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.
19. 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.
20. 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.