Hybridizing an Immune Artificial Algorithm with Epsilon Constrained Method

Victoria S. Aragón and Susana C. Esquivel

Laboratorio de Investigación y Desarrollo en Inteligencia Computacional**
Universidad Nacional de San Luis
Ejército de los Andes 950 - (5700) San Luis, ARGENTINA

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 algorithm is called TCEC (T-Cell Epsilon Constrained) due to it is increased with epsilon constrained method, for solving constrained (numerical) optimization problems. We validate our proposed approach with a set of 36 test functions provided for the CEC 2010 competition. We indirectly compare our results with respect to a version of the differential evolution algorithm. Our results show that TCEC can found feasible solutions on almost test functions with 10 and 30 decision variables.

Keywords: Artificial Immune System, Constrained Optimization Problem, Epsilon Constrained Method.

1 Introduction

Over the last years, a bio-inspired system has called 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 computational point of view, such as: uniqueness, pattern recognition, diversity, fault tolerance, learning and memory, self-organization, robustness, cooperation between different layers, among others [12]. Thus, these characteristics and a well-known about the functionality of the NIS are excellent motivations to develop Artificial Immune Systems (AIS) to handle 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 eliminates the binary

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

representation present in its original version and also includes the ϵ constrained method in order to find in a faster way feasible solutions for a set of test functions.

The remainder of the paper is organized as follows. Section 2, defines the problem we want to solve. Section 3 presents a brief revision of the artificial immune systems existing. In Section 4 describes the proposed algorithm. Section 5 shows our experimental setup and results, for last in 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

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

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 restricted for 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. We called $\phi(X)$ to the maximum value of all constraints the problem has for X solution, i. e., $\phi(X) = max\{max_j\{0, g_j(X)\}, max_k \mid h_k(X) \mid\}$.

2.1 The ϵ Constrained Method

The ϵ constrained method converts a constrained optimization problem into an unconstrained one by replacing the order relation in direct search methods with the ϵ level comparison. In this paper, these comparisons are defined as a lexicographic order in which the maximum value of all constraints is more important than the objective value, i. e., the feasibility of an solution X is more important than the minimization of $f(X)$ [14]. Thus, the problem is redefined as:

minimize

$$f(X) \tag{2}$$

subject to $\phi(X) \leq \epsilon$

3 Previous Work

According to [7] the main Artificial Immune System models are: Negative Selection [6], Clonal Selection [11] and Immune Network Models [8] and [7]. 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. Besides, we could not find in current literature an artificial

immune system which has been hybridizing with the well known method epsilon constrained. Even when some others approaches have been hybridizing with this techniques as [13], [14] and [10], among others.

4 Proposed Algorithm Based on TCELL

Here, an immune algorithm based on the immune responses mediated by the T cells is developed. It is inspired on the TCELL model [1]. It is called TCEC (Constrained T-Cell with Epsilon Constrained) and modified and add to it the well known epsilon constrained method to solve constrained optimization problems.

First approaches based on TCELL model were used to solve static optimization problems, dynamic problems, constrained problems and dynamic constrained problems [3], [1] and [4]. Basically, TCEC 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. It means, the cells go through positive and negative selection and then they will be activated (proliferation and differentiation processes). Each of these processes are defined next.

TCEC works over three populations, corresponding to the groups in which the T-cells are divided: (1) Virgin Cells (VC), (2) Effector Cells (EC) and (3) Memory Cells (MC). Each population is composed by a set of T-cells, where each cell is composed by: 1) a TCR_r (represented by a vector of real numbers, they identify the decision variables of the problem), 2) a proliferation level (number of clones assigned to the cell when it has to proliferate) and 3) a differentiation level (number of decision variables that will be changed when it has to differentiate).

Virgin Cells (VC) do not suffer an activation process. The goal of effector cells (EC) and the memory cells (MC) are to explore in a global way the search space and to explore the neighborhood of the best found solutions, respectively.

Positive selection eliminates 10% of the worst effector cells and negative selection eliminates effector 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 Euclidean distance) and the worst between them is eliminated. This process reduces the effector's population size.

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)[1] and 2) the value of the objective function (only if the cell is feasible).

Only effector cells use epsilon constrained method, on equality constraints, in order to not remove some infeasible solutions from population.

In this work we control the ϵ level according to [14].

$$\epsilon(0) = \phi(x_\theta) \tag{3}$$

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

where x_θ is the θ-th cell from EC. ϵ level is adjusted to be a small value (0.0001) at iteration 1000. Before EC be activated ϵ for EC is updated as follow:

$$cp = (-5 - log(epsilon(0))/log(0.05) \tag{4}$$

$$\epsilon_{EC} = epsilon(0) * (1.0 - (iteration/1000.0))^{cp} \tag{5}$$

$$cp = 0.3 * cp + (0.7 * 3.0) \tag{6}$$

term cp, proposed by [14], is decreased in order to search better objective values.

We consider a ce_i cell is better than a ce_j cell if 1) TCR_r's ce_i is feasible and TCR_r's ce_j is infeasible, 2) both cells have feasible TCR_rs but objective function value's ce_i is lower than objective function value's ce_j and 3) both cells have infeasible TCR_rs but sum_res' ce_i is lower than sum_res' ce_j. This criterion is used to perform population sort.

4.1 Effector Cells Activation

Only some effector cells through activation process per iteration (U(1,| EC| /2) [2] [3]. In order to not bias the search to the best found solutions on each iteration, with a 80% of probability the best solutions are chosen randomly.

Now, when an effector cell is chosen to be activated, called ce_i, this implies the random selection of a set of potential activator (or stimulating) cells, this set is composed by some members, given by U(| EC| /2,| EC|), this means at least half of cells are chosen to be potential activator of ce_i.

The closest or farthest cell to ce_i, according to the TCR_r in the set, is chosen to become the stimulating cell, say ce_j. Then, ce_i proliferates and differentiates. The proliferation level of each stimulated cell, ce_i, depends on its stimulating cell. If ce_j is the farthest cell to ce_i then ce_i will receive, as clones, U(1, MAX_CLONES), otherwise ce_i will receive MAX_CLONES. Even when all this clones are assigned to each cell, proliferation process stops when, after the differentiation of a clone, this is better than its original cell and this clone is inserted into the effector population. This is aimed by not consume so many function evaluations per one cell, if it is possible.

By the other hand, the differentiation level for cell ce_i too is related to its stimulating cell (ce_j). If ce_j is the farthest to ce_i then this level is U(30% dv, 50% dv), otherwise is U(10% dv, 30% dv). [4]

These level combination are motivated to modify less clones in an intense way, when they are far from the original cell, and more clones in a soft way, when they are close to the original cell. In order to explore the search space but exploit the promising solutions found with the aid of the epsilon constrained method.

[2] $U(x, y)$ returns a random number with a uniform distribution in the range (x,y).
[3] | EC| means EC size.
[4] dv means the number of decision variables of the problem.

Once different levels are settled the decision variables to be changed are chosen in a randomly way. In this step we consider which cell is better(ce_i or ce_j), in order to drive the search to the best explore areas. Thus, each variable to be changed is modified (with 95% probability) according to add or subtract a random value r, $x^{'} = x \pm r$ where x and $x^{'}$ are the original and the differentiated decision variables, respectively. Otherwise, a random number with a uniform distribution is signed to $x^{'}$. 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 $r = U(0, (x - ll_x) * factor)$ is subtracted from x; otherwise, $r = U(0, (lu_x - x) * factor)$ is added to x. lu_x and ll_x are the upper and lower bounds of x, respectively. $factor = 1 - \frac{dist}{max\_dist}$ regulated how much is influenced the stimulated from the stimulating, $dist$ is the Euclidean distance between the cells and max_dist is the maximum distance between two points into the search space. If, after add or subtract r to x, $x^{'}$ is not in the allow range then $U(ll_x - x)$ or $U(x - lu_x)$ is assigned to $x^{'}$, respectively.

4.2 Memory Cells Activation

Memory cells proliferate according to a maximum number of clones and their differentiation level is given by random[1, dv]. Both levels are independent from the other memory cells. All clones are evaluated in order to find the best of them to replace its original cell if it is better than this.

Here, each variable to be changed is chosen in a random way and it is modified using the following equation: $x^{'} = x \pm r$ where $r = \left(\frac{U(0, lu_x - ll_x)}{iter+1}\right)^{U(0,1)}$, x and $x^{'}$ are the original and the differentiated decision variables, respectively. 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 add or subtract r to x, $x^{'}$ is not in the allow range then $U(ll_x - x)$ or $U(x - lu_x)$ is assigned to $x^{'}$, respectively.

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

TCEC Algorithm

```
Initialize_VC();
Calculate_Constraint_VC();
Calculate_Epsilon0_VC()
Evaluate_VC();
Sort_VC();
Assign_Proliferation();
Inserte_VC_into_EC();
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 rep1
    Update_Epsilon_EC();
    Activate_EC();
    Sort_EC();
  endfor
  Insert_CDs_en_MC();
  for i=1 to rep2
    Activate_MC();
  endfor
  Sort_MC();
od
Statistics();
```

The algorithm works in the following way. At the beginning, the TCR_r from the virgin cells are initialized in a random way. Thus, for each TCR_r of a virgin cell the constraints are calculated. Then, $\epsilon(0)$ is calculated according to Eq. 3 and virgin cells are evaluated and sorted. The proliferation levels are assigned. The best virgins cells are inserted into EC. Each effector cell will inherit the proliferation level of the virgin cell which received the TCR.

The negative and positive selections are applied to effector population.

A maximum number of objective function evaluations is allowed. Then the actions are: update epsilon for EC according to Eq. 4 to 6, to activate the EC population $rep1$ times; in other words, to perform proliferation and differentiation of selected cells from EC. The first clone better than the original cell is passed to the next iteration. Then, these cells are sorted.

If MC is empty the best solutions from EC are inserted in MC. Otherwise, best solution from EC is used to replace the worst solution in MC. In any case, first, the solutions are reevaluated with epsilon=0.0001. Next, the cells from MC are activated a certain (predefined) number of times, $rep2$.

Main differences between the original version of the algorithm based on TCELL model and TCEC include: 1) TCEC eliminates a population called CD4 which uses binary representation for its TCRs and 2) TCEC incorporates epsilon constrained method.

5 Numerical Experiments and Results

In order to validate TCEC we test it with 36 constrained problems. They were proposed on CEC 2010 [9]. Eighteen test problems have 10 decision variables (10D) and the others eighteen have 30 decision variables (30D). In problems with equality constraints, these are relaxed and converted to inequality constraints according to $\mid h_k(X) \mid \leq \epsilon$. Some preliminary runs were performed. We used the following parameters: population sizes, $\mid VC \mid = 100$, $\mid EC \mid = 5$, $\mid MC \mid = 2$. rep=10 and $rep1$=100 (for 10D problems C05, C06, C09, C10, C11 and C17 $rep1$=1000 was required and, for C11 and C17 $\mid MC \mid = 5$ was used). 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 are best, worse and average values. The maximum number of functions evaluations is 2×10^4 for 10D problems and 6×10^4 for 30D problems. Our results, are indirectly compared with respect to *jDE-Mut* [5], It performs 25 independent runs and the same number of function evaluations. It is worth noting that *jDE-Mut* calculates the average measure according to feasible and infeasible solutions. The results are showed in tables 1 and 2.

We can see that our algorithm was able to reach feasible solutions for almost all test function, except for C03 and C11 both with has 30D. It seems that TCEC is not able to find feasible solutions when the objective function is non separable or rotated and the constraint equality is non separable.

For 10D problems (see Table 1), comparing our performance with respect to *jDE-Mut* our TCEC obtained better results in twelve test problems (C02, C04 to C06, C11 to C18). But, TCEC was outperformed in remaining five test problems. For C01, TCEC has a better performance considering the worst and media values and *jDE-Mut* got a better value on best measure respect to TCEC, though very small. It worth noting *jDE-Mut* got feasible solution on C03 when TCEC can not do it. For 30D problems (see Table 2), comparing our performance with respect to *jDE-Mut* our TCEC obtained better results in nine test problems (C02, C04 to C06, C12, C14, C16 to C18). TCEC was outperformed in seven test problems (C01, C07 to C10, C13 and C15). Any algorithm could found feasible solutions for C03 and C11.

Regardless of the number of decision variables *jDE-Mut* does not found any feasible solutions for (C04 to C06, C11 and C12) while TCEC does, but TCEC could not found feasible solutions on each run for these five functions, the same happens for C02. Last, for C09, C10, C16, C17 and C18 (with 30D) TCEC could not found feasible solutions on each run. Even when TCEC outperforms *jDE-Mut* on 21 of the 36 problems our approach shows a big standard deviation on some of the functions.

6 Conclusions and Future Work

This paper presents a modified version an Artificial Immune System based on T-Cell Model for solving constrained optimization problems which includes the epsilon constrained method in one of the three algorithm's populations in order to find feasible solutions. The algorithm is called TCEC.

Our proposed algorithm was found to be able to converge to feasible solutions in almost all cases tested, only it fails in 2 of the 36 test functions. And it shows to be competitive over the benchmark used when it is compared with a differential evolution algorithm. Even, TCEC found feasible solutions in 10 test functions when the algorithm we compare fails.

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 differentiation processes in order to find more

Table 1. Function Values Achieves for 10D Problems. The values in **bold** indicate which algorithm found the best value. RF% means percentaje of run where a feasible solution was found. - indicates any feasible solution was found. NotRep means authors do not report this value.

Problem	TCEC			*jDE-Mut*		
	Best (RF%)	Worst	Mean (Std.)	Best (RF%)	Worst	Mean (Std.)
C01	-7.45e-01 (100%)	**-6.91e-01**	**-7.35e-01** (1.33e-02)	**-7.46e-01** (100%)	-6.75e-01	-7.34e-01 (1.58e-02)
C02	**-2.20e+00** (63.3%)	**4.48e+00**	**-3.33e-02** (1.88e+00)	1.33e-01 (NotRep)	-	1.45e+00 (1.27e+00)
C03	- (0%)	-	- (-)	**1.56e-05** (NotRep)	-	**9.57e+00** (4.20e+00)
C04	**5.23e-02** (50%)	**1.40e+01**	**1.21e+01** (4.91e+00)	- (0%)	-	- (-)
C05	**-4.62e+02** (33.3%)	**1.49e+02**	**-2.89e+02** (2.33e+02)	- (0%)	-	- (-)
C06	**-5.74e+02** (30%)	**-5.37e+02**	**-5.64e+02** (1.18e+01)	- (0%)	-	- (-)
C07	3.65e+01 (100%)	3.77e+03	5.80e+02 (8.19e+02)	**8.71e-02** (100%)	**1.92e+01**	**4.17e+00** (4.57e+00)
C08	7.32e+00 (100%)	1.38e+03	1.86e+02 (2.82e+02)	**1.10e-04** (100%)	**1.08e+02**	**1.52e+01** (2.86e+01)
C09	1.42e+02 (100%)	**3.45e+10**	1.51e+09 (6.42e+09)	**1.85e-08** (NotRep)	-	**7.28e+03** (3.62e+04)
C10	2.00e+02 (100%)	**5.51e+10**	2.23e+09 (1.01e+10)	**4.01e-06** (NotRep)	-	**7.79e+04** (3.20e+05)
C11	**7.57e-05** (23.3%)	**1.39e-03**	**8.38e-04** (4.73e-04)	- (0%)	-	- (-)
C12	**-5.64+02** (96.6%)	**4.06+02**	**-1.73+01** (1.46e+02)	- (0%)	-	- (-)
C13	**-6.84e+01** (100%)	**-6.28e+01**	**-6.74e+01** (1.64e+00)	-6.79e+01 (100%)	-5.97e+01	-6.59e+01 (2.36e+00)
C14	**4.24e+00** (100%)	**3.88e+06**	**1.30e+05** (7.08e+05)	1.90e+09 (100%)	4.17e+12	3.40e+11 (8.83e+11)
C15	**3.35e+09** (100%)	**4.10e+11**	**9.81e+10** (1.05e+11)	4.33e+09 (NotRep)	-	3.67e+13 (6.45e+13)
C16	**7.46e-03** (100%)	**1.18e+00**	**3.38e-01** (3.91e-01)	6.69e-01 (NotRep)	-	9.64e-01 (1.09e-01)
C17	**6.23e-05** (100%)	**2.88e+00**	**1.02e+00** (9.58e-01)	6.70e+01 (NotRep)	-	2.65e+02 (1.82e+02)
C18	**1.16e-05** (100%)	**5.30e+02**	**5.73e+01** (1.08e+02)	1.77e+03 (NotRep)	-	7.93e+03 (5.38e+03)

Table 2. Function Values Achieves for 30D Problems. The values in **bold** indicate which algorithm found the best value. RF% means percentaje of run where a feasible solution was found. - indicates any feasible solution was found. NotRep means authors do not report this value.

Problem	TCEC			*jDE-Mut*		
	Best	Worst	Mean (Std.)	Best	Worst	Mean (Std.)
C01	-7.89e-01 (100%)	-6.98e-01	-7.47e-01 (2.5e-02)	**-8.1e-01** (100%)	**-7.17e-01**	**-7.93e-01** (-7.93e-01)
C02	**-1.34e+00** (100%)	**4.85e+00**	**1.28e+00** (2.06e+00)	1.14e+00 (NotRep)	-	3.72e+00 (1.02e+00)
C03	- (0%)	-	- (-)	- (0%)	-	- (-)
C04	**3.80e-04** (100%)	**1.93e+01**	**9.68e+00** (1.03e+01)	- (0%)	-	- (-)
C05	**-7.60e+00** (63.3%)	**5.51e+02**	**2.67e+02** (1.51e+02)	- (0%)	- (-)	-
C06	**-3.13e+02** (100%)	**5.78e+02**	**2.15e+02** (2.68e+02)	- (0%)	-	- (-)
C07	2.30e+02 (100%)	1.79e+04	2.88e+03 (5.10e+03)	**1.25e+01** (100%)	**8.98e+01**	**4.12e+01** (3.07e+01)
C08	2.89e+01 (100%)	1.33e+04	2.11e+03 (3.442e+03)	**2.05e+01** (100%)	**2.59e+01**	**1.43e+02** (2.49e+02)
C09	2.77e+08 (23.3%)	**1.79e+11**	2.63e+10 (6.76e+10)	**7.47e-01** (NotRep)	-	**6.88e+09** (3.44e+10)
C10	8.53e+08 (40%)	**4.82e+12**	5.59e+11 (1.40e+12)	**3.14e+01** (NotRep)	-	**5.87e+01** (-)
C11	- (0%)	-	- (-)	- (0%)	-	- (-)
C12	**-1.97e-01** (80%)	**1.50e+01**	**1.12e+00** (3.46e+00)	- (0%)	-	- (-)
C13	-6.49e+01 (100%)	-5.55e+01	-5.95e+01 (2.47e+00)	**-6.51e+01** (100%)	**-5.94e+01**	**-6.22e+01** (1.44e+00)
C14	**2.97e+01** (100%)	**3.56e+09**	**1.80e+08** (6.71e+08)	1.58e+12 (100%)	3.27e+13	1.07e+00 (8.12e+12)
C15	3.29e+11 (100%)	5.24e+12	2.41e+12 (1.29e+12)	**3.99e+09** (100%)	**2.33e+13**	**2.16e+12** (5.22e+12)
C16	**1.70e-01** (83.3%)	**1.18e+00**	**8.93e-01** (3.01e-01)	1.02e+00 (NotRep)	-	1.02e+00 (2.52e-02)
C17	**2.79e-01** (23.3%)	**1.65e+02**	**5.37e+01** (5.77e+01)	1.50e+03 (NotRep)	-	1.59e+03 (4.35e+02)
C18	**1.77e+02** (100%)	**3.31e+04**	**6.80e+03** (1.02e+04)	6.97e+03 (NotRep)	-	3.28e+04 (1.33e+04)

quickly feasible solutions and also to improve the quality of the found feasible solutions. Besides, we wish to realize a statistical analysis about the influence of the parameters over the performance of TCEC.

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. A modified version of a t-cell algorithm for constrained optimization problems. *International Journal for Numerical Methods in Engineering*, 84(3):351–378, 2010.
4. 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.
5. J. Brest, A. Zamuda, I. Fister, B. Boskovic, and M. S. Maucec. Constrained real-parameter optimization using a differential evolution algorithm. In *IEEE SSCI2011 symposium series on computational intelligence*, pages 9 –16, 2011.
6. 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.
7. Simon M. Garrett. How do we evaluate artificial immune systems? *Evol. Comput.*, 13(2):145–177, 2005.
8. N. K. Jerne. The immune system. *Scientific American*, 229(1):52–60, 1973.
9. R. Mallipeddi and P. N. Suganthan. Problem definitions and evaluation criteria fir the cec 2010 competition on constrained real-parameter optimization. Technical report, Nanyang Technological University, Singapore, 2010.
10. E. Mezura-Montes, M. Damian-Araoz, and O. Cetina-Dominguez. Smart flight and dynamic tolerances in the artificial bee colony for constrained optimization. In *IEEE Congress on Evolutionary Computation (CEC2010)*, pages 4118–4125, Barcelona, Spain, 2010.
11. 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.
12. Leandro Nunes de Castro and Jonathan Timmis. *Artificial Immune Systems: A New Computational Intelligence Approach*. Springer-Verlag, New York, 2002.
13. Tetsuyuki Takahama and Setsuko Sakai. Fast and stable constrained optimization by the constrained differential evolution. In *Pacific Journal of Optimization*, page 261282, 2009.
14. Tetsuyuki Takahama and Setsuko Sakai. Constrained optimization by the epsilon constrained differential evolution with an archive and gradient-based mutation. In *IEEE Congress on Evolutionary Computation (CEC2010)*, pages 1680–1688, Barcelona, Spain, 2010.