Handling Constraints in Global Optimization using an Artificial Immune System

Nareli Cruz-Cortés, Daniel Trejo-Pérez and Carlos A. Coello Coello

CINVESTAV-IPN, Evolutionary Computation Group
Depto. de Ingeniería Eléctrica, Sección de Computación
Av. Instituto Politécnico Nacional No. 2508 Col. San Pedro Zacatenco,
México, D. F. 07360, MEXICO
{nareli,dtrejo}@computacion.cs.cinvestav.mx,
ccoello@cs.cinvestav.mx

Abstract. In this paper, we present a study of the use of an artificial immune system (CLONALG) for solving constrained global optimization problems. As part of this study, we evaluate the performance of the algorithm both with binary encoding and with real-numbers encoding. Additionally, we also evaluate the impact of the mutation operator in the performance of the approach by comparing Cauchy and Gaussian mutations. Finally, we propose a new mutation operator which significantly improves the performance of CLONALG in constrained optimization.

1 Introduction

Many bio-inspired algorithms (particularly evolutionary algorithms) have been very successful in the solution of a wide variety of optimization problems [3]. However, all of these approaches (including evolutionary algorithms and artificial immune systems), when used for numerical optimization, can be seen as unconstrained search techniques. This means that they require a suitable mechanism to incorporate constraints, such that they can deal with the general nonlinear optimization problem.

Within evolutionary algorithms (EAs), external penalty functions have been the most popular mechanism adopted to incorporate constraints into the fitness function [17]. The idea of an external penalty function is to "punish" (or penalize) a solution for being infeasible by increasing its fitness value (when solving a minimization problem). Despite their popularity, penalty functions have several problems, from which the main one relates to the difficulties to define accurate penalty factors. A large penalty value discourages the exploration of the infeasible region since the very beginning of the search process (this may cause difficulties when dealing, for example, with a disjoint feasible region). On the other hand, if the penalty value is too low, a lot of the search time will be spent exploring the infeasible region because the penalty will be negligible with respect to the objective function.

Recently, several researchers have proposed constraint-handling techniques for EAs which avoid the use of a penalty function or do not require any fine-tuning of the penalty factors [11, 16, 7, 9]. Such approaches have been found to outperform traditional penalty functions and can handle all types of constraints (linear, nonlinear, equality, inequality).

The main motivation of the work presented in this paper was to explore the capabilities of an artificial immune system in the context of constrained global optimization. For that sake, we decided to adopt the algorithm based on the clonal selection principle (called CLONALG) which is described in [4, 6]. CLONALG was proposed as a learning algorithm particularly well-suited for solving pattern recognition and multimodal optimization problems. However, as far as we know, it hasn't been properly validated in the context of constrained global optimization.

CLONALG is a population-based algorithm and its only variation operator is mutation. Evidently, the main search power of CLONALG relies on this mutation operator and therefore, such operator became the main focus of our study.

There is well-documented evidence (in the specialized literature) of the superiority of evolution strategies (over genetic algorithms) in constrained optimization [16]. Also, we were aware of the proposal by Yao and Liu [18] of adopting Cauchy-distributed random numbers for an evolution strategy, instead of the traditional Gaussian distribution. Yao and Liu [18] found that this probability distribution allowed the mutation operator of an evolution strategy to behave as a sort of crossover operator, thus improving the performance of the algorithm in (unconstrained) numerical optimization problems (including multimodal functions). Apparently, this behavior was due to the fact that Cauchy-distributed random numbers allow relatively coarse-grained steps, which contrast with the fine-grained steps of the traditional Gaussian-distributed random numbers.

Thus, being aware of this work, we decided to incorporate Cauchy-distributed random numbers into CLONALG. We hypothesized that this would considerably improve the performance of CLONALG in constrained optimization (with respect to the use of Gaussian mutations). However, the results (as we will see later on) were rather disappointing and led us to propose our own mutation operator which turned out to be better than any of the two other operators initially adopted. Finally, we also evaluated CLONALG with binary encoding, in order to assess the impact of the representation in the performance of the approach (in the context of constrained optimization).

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 related work. In Section 4, we describe the modifications done to CLONALG so that it can handle constraints. In Section 5, we present our experiments adopting a binary representation (both with and without Gray coding). In Section 6, we report our experiments adopting real-numbers representation (adopting different mutation operators). In Section 7, some changes to CLONALG's mutation operator are proposed, our results are presented and they are discussed. Finally, in Section 8, our conclusions and some possible paths for future research are provided.

2 Statement of the Problem

The problem that we want to solve is the general nonlinear programming problem which is defined as follows:

$$\text{Find } \boldsymbol{x} \text{ which optimizes } f(\boldsymbol{x}) \tag{1}$$

subject to:

$$g_i(\boldsymbol{x}) \leq 0, \;\; i = 1, \ldots, n \tag{2}$$

$$h_j(\boldsymbol{x}) = 0, \;\; j = 1, \ldots, p \tag{3}$$

where $\boldsymbol{x}$ is the vector of solutions (or decision variables) $\boldsymbol{x} = [x_1, x_2, \ldots, x_r]^T$, n is the number of inequality constraints and p is the number of equality constraints (in both cases, constraints could be linear or nonlinear).

3 Previous Related Work

We were able to find very few papers in which the main focus was the solution of constrained global optimization problems using an artificial immune systems. Such work is briefly described next.

Hajela and Yoo [19, 8] proposed a hybrid between a Genetic Algorithm (GA) and an artificial immune system, aiming to solve constrained optimization problems. In this approach, the authors adopted 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 is run for a few generations, and uses as its fitness function a Hamming distance (binary encoding was adopted for the GA) so that the antibodies are evolved to become "very similar" (at the genotypic level) to the antigens, without becoming identical. The interesting effect of this evolution was that the infeasible individuals would normally become feasible. This approach was tested with some structural optimization problems.

Kelsey and Timmis [10] proposed an immune inspired algorithm based on the clonal selection theory to solve multimodal optimisation 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 [2] proposed an extension of Hajela and Yoo's algorithm. In this proposal, no penalty function is required (as in the original approach), and some extra mechanisms are defined to allow the approach to work in cases in which there are no feasible solutions in the initial generation. Additionally, the authors proposed a parallel version of the algorithm and validated it using some standard test functions reported in the specialized literature.

Balicki's proposal [1] is very similar to the two previous approaches. 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 [13, 12] proposed an algorithm (called CMOIA, or Constrained Multi Objective Immune Algorithm) for solving constrained multi-objective optimization problems. In this case, the antibody's population is composed by 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 (note that this IL function is actually a penalty function). 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 through six test functions and two structural optimization problems.

4 Clonal Selection Algorithm for Constrained Optimization

Nunes and Von Zuben [4, 6] proposed the CLONALG algorithm, which is inspired in the clonal selection theory of the immune system. CLONALG was used by its authors to solve pattern recognition and multimodal optimization problems.

Next, we will describe the main elements that comprise the general framework of any biological-inspired system as described in [5] and considering the context of constrained optimization:

- **Representation of the components**: *Antigens* are represented by the objective function $f(\boldsymbol{x})$ that we want to optimize (minimize or maximize) and *antibodies* are represented by the variables of the problem $(\boldsymbol{x})$ which are potential solutions.
- **Mechanisms to evaluate the interaction among individuals and their environment**: antibody's affinity corresponds to the evaluation of the objective function given by the antigen.
- **Adaptation Procedures**: The clonal selection theory of the immune system.

Now, we will briefly describe the CLONALG algorithm [6]:

1. Generate j antibodies randomly.
2. Repeat a predetermined number of times:
   (a) Determine the affinity of each antibody (*Ab*). This affinity corresponds to the evaluation of the objective function.
   (b) Select the n highest affinity antibodies.
   (c) The n selected antibodies will be cloned proportionally to their affinities, generating a repertory C of clones: the higher the affinity is, the higher becomes the number of clones generated for each of the n selected antibodies.
   (d) The clones from C are subject to a hyper-mutation process inversely proportional to their antigenic affinity: the higher the affinity, the smaller the mutation rate.
   (e) Determine the affinity of the mutated clones C.
   (f) From this set C of clones and antibodies (*Ab*), select the j highest affinity clones to compose the new antibodies' population.
   (g) Replace the d lowest affinity antibodies by new individuals generated at random.
3. End repeat

The number of clones generated from the n selected antibodies is given by:

$$Nc = \sum_{i=0}^{n} round\left(\frac{\beta * j}{i}\right) \tag{4}$$

where Nc is the total number of clones, and β is a multiplier factor (generally equal to 1).

In order to apply CLONALG to constrained optimization problems, we introduced the following changes:

- In step 2.(a), it is necessary that each antibody evaluates the objective function, and the constraints of the problem in order to know if it is a feasible solution or not.
- In step 2.(b), the antibodies' affinity is defined not only by the objective function value, but also based on feasibility or infeasibility, considering that feasible individuals must have a higher affinity value. Within this group, the best individuals are those whose objective function value is best (i.e., the larger value if we are maximizing). Within the group of infeasible individuals, those having the lowest constraint violation quantity will obtain the highest affinity values (since they are closest to the feasible region).
- In step 2.(f), when the individuals that will make it for the next generation are determined, it is necessary to ensure that at least q infeasible individuals will survive (q is a user-defined parameter). This is because we want to promote a reasonable diversity right in the boundary between the feasible and the infeasible region (this is done because it is known that the most difficult constrained optimization problems for EAs are those in which the global optimum is located precisely in the boundary between the feasible and the infeasible regions).
- In step 2.(g), the criterion that determines the individuals who will be replaced is driven by feasibility criteria as well. That means that if infeasible individuals become a majority, more of them will be replaced and vice versa.

In order to assess the performance of CLONALG in constrained optimization, we conducted a set of experiments using different representations and mutation operators. For these experiments, we adopted the benchmark originally proposed in [15] and extended in [16]. The test functions chosen contain characteristics that are representative of what can be considered "difficult" global optimization problems for an evolutionary algorithm (or any other bio-inspired algorithm used for global optimization). The mathematical description of these test functions can be found in [16].

To get an estimate of how difficult is to generate feasible points through a purely random process, we computed the ρ metric for the 13 test functions of the benchmark (as suggested by Michalewicz and Schoenauer [15]) using the expression:
$\rho = |F|/|S|$, where $|S|$ is the number of random solutions generated ($S = 1,000,000$ in our case), and $|F|$ is the number of feasible solutions found (out of the total $|S|$ solutions randomly generated). The values of ρ for each of the functions chosen are shown in Table 1.

CLONALG was compared with respect to two approaches that are representative of the state-of-the-art in constrained optimization: Stochastic Ranking [16] and the Adaptive Segregational Constraint Handling Evolution Strategy (ASCHEA) [9].

The versions that were evaluated are the following:

- Binary representation
  - Standard
  - Gray coding

Problem	n	Type of function	ρ	LI	NI	LE	NE
g01	13	quadratic	0.0003%	9	0	0	0
g02	20	nonlinear	99.9973%	1	1	0	0
g03	10	nonlinear	0.0026%	0	0	0	1
g04	5	quadratic	27.0079%	0	6	0	0
g05	4	nonlinear	0.0000%	2	0	0	3
g06	2	nonlinear	0.0057%	0	2	0	0
g07	10	quadratic	0.0000%	3	5	0	0
g08	2	nonlinear	0.8581%	0	2	0	0
g09	7	nonlinear	0.5199%	0	4	0	0
g10	8	linear	0.0020%	3	3	0	0
g11	2	quadratic	0.0973%	0	0	0	1
g12	3	quadratic	4.7697%	0	9^3	0	0
g13	5	nonlinear	0.0000%	0	0	1	2

Table 1. Values of ρ for the 13 test problems. n=number of decision variables, LI=number of linear inequalities, NI=number of nonlinear inequalities, LE=number of linear equalities and NE=number of nonlinear equalities.

– Real-numbers representation
  - Self-adaptive mutation using Gaussian distribution
  - Self-adaptive mutation using Cauchy distribution
  - Uniform mutation without self-adaptation

5 Binary Representation

The CLONALG algorithm with the extensions for handling constrained optimization problems, was first implemented using a binary representation. Since other researchers have reported advantages of using Gray codes when dealing with numerical optimization problems (and adopting genetic algorithms) [14], we decided to try both versions: normal binary encoding and binary encoding with Gray codes.

For these experiments, we adopted the following parameters:

– Number of antibodies j = 50.
– Minimum number of infeasible solutions that must survive q=2.
– Type of mutation: uniform (same as in the simple genetic algorithm).
– Mutation rate: Clones are sorted by their affinity values in a descendent manner. The mutation operator was implemented using a small value at the beginning and then we increased it until a certain predetermined value was reached. The initial mutation rate was= $(1/len)$, where len is the length of the binary string. The final mutation rate was set to 0.3.
– Percentage d of replaced antibodies: 0.20.
– Total number of objective function evaluations: 238,750.

Test Function	Optimal	Best	Mean	Worst	Std. Dev
g01*	-15.0	-14.8686	-14.660287	-12.789464	0.501459
g02	0.803619	0.775589	0.749575	0.683894	0.025699
g03*	1.0	0.99891	0.97078	0.92849	0.024912
g04	-30665.539	-30650.00697	-30460.85416	-30366.98548	96.1407
g05	5126.498	INF	INF	INF	INF
g06	-6961.814	-6921.48749	-6248.9307	-6182.9937	181.4024
g07	24.306	24.80870	30.8661	35.4455	2.4353
g08	0.095825	0.095825	0.093398	0.09313	0.00080
g09	680.630	684.12886	704.87263	753.22103	17.58436
g10	7049.25	INF	INF	INF	INF
g11	0.75	0.750295	0.865079	1.567670	0.22056
g12	1.0	0.999996	0.907750	0.725285	0.077762
g13	0.053950	INF	INF	INF	INF

Table 2. Results obtained by CLONALG using standard binary representation. INF means that the algorithm converged to an infeasible solution. In the functions marked with *, the approach converged to a feasible solution only in 75% of the runs.

Test Function	Optimal	Best	Mean	Worst	Std. Dev.
g01	-15.0	-15.0	-15.0	-15.0	0.0
g02	0.803619	0.760753	0.700945	0.562943	0.049646
g03	1.0	1.0	0.999688	0.997992	0.000579
g04	-30665.539	-30665.504	-30662.678	-30658.778	1.946
g05*	5126.498	INF	INF	INF	INF
g06	-6961.814	-6961.813	-6961.813	6961.810	0.000552
g07	24.306	24.945	27.017	30.007	1.709
g08	0.095825	0.095825	0.095825	0.095825	0.0
g09	680.630	680.727	681.649	682.853	0.599
g10*	7049.25	7451.54	8344.18	10509.34	962.58
g11	0.75	0.75	0.76	0.79	0.011
g12	1.0	1.0	1.0	1.0	0.0
g13**	0.053950	0.059553	0.059215	0.062709	0.003000

Table 3. Results obtained by CLONALG using binary representation with Gray coding. INF means that the algorithm converged to an infeasible solution. In the functions marked with *, the approach converged to a feasible solution only in 50% of the runs. The use of ** indicates cases in which the algorithm converged to a feasible solution only in 15% of the runs.

The parameters used to compute the number of clones (Eq. (4)) were the following: $\beta = 1$ and $n = j$.

Tables 2 and 3 summarize the statistical results obtained by our binary versions of CLONALG (with and without Gray coding, respectively) over 30 independent runs. Clearly, the version that uses Gray coding outperforms its traditional binary counterpart. When standard binary representation is used, the algorithm is not capable of finding feasible solutions in functions g5, g10 and g13. For all the other functions, a reasonable approximation to the optimal value is attained.

When Gray coding is adopted, in 10 of the 13 functions, the algorithm is capable of reaching the optimal value. However, in functions g5, g10, and g13, the algorithm has problems even for reaching the feasible region, as indicated before.

Test Function	Optimal	Best	Mean	Worst
g01	-15.0	-15.0	-14.84	N.A.
g02	0.803619	0.785	0.59	N.A.
g03	1.0	1.0	0.99989	N.A.
g04	-30665.5	-30665.5	-30665.5	N.A.
g05	5126.4981	5126.5	5141.65	N.A.
g06	-6961.814	-6961.81	-6961.81	N.A.
g07	24.306	24.3323	24.6636	N.A.
g08	0.095825	0.095825	0.095825	N.A.
g09	680.63	680.630	680.641	N.A.
g10	7049.33	7061.13	7497.434	N.A.
g11	0.75	0.75	0.75	N.A.
g12	-1.0	N.A.	N.A.	N.A.
g13	0.05395	N.A.	N.A.	N.A.

Table 4. Results obtained by ASCHEA [9] performing 1,500,000 objective function evaluations. N.A.=Not available

To have an idea of how competitive are the results produced by our two binary versions of CLONALG, we present in Table 4 the results produced by ASCHEA [9]. These results were obtained with 1,500,000 objective function evaluations (let's keep in mind that CLONALG performed only 238,750 evaluations). It can be clearly seen that CLONALG (in its two versions) is outperformed by ASCHEA in most cases. Note however that ASCHEA does not report results for g12 and g13.

Table 5 the results produced by Stochastic Ranking [16]. These results were obtained with 350,000 objective function evaluations. As can be seen in Table 5, Stochastic Ranking obtained better results for six functions, and CLONALG outperformed Stochastic Ranking only in one test function.

6 Real-Numbers Representation

In this Section we present our experiments using a version of CLONALG with real-numbers representation. In [6], the authors proposed to use self-adapting mutation pa-

Test Function	Optimal	Best	Mean	Worst	Std. Dev.
g01	-15.0	-15.0	-15.0	-15.0	0.0E+00
g02	0.803619	0.803515	0.7858	0.726288	2.0E-02
g03	1.0	1.0	1.0	1.0	1.9E-04
g04	-30665.539	-30665.539	-30665.539	-30665.539	2.0E-05
g05	5126.498	5126.497	5128.881	5142.472	3.5E+00
g06	-6961.814	-6961.814	-6875.940	-6350.262	1.6E+02
g07	24.306	24.307	24.374	24.642	6.6E-02
g08	0.095825	0.095825	0.095825	0.095825	2.6E-17
g09	680.63	680.630	680.656	680.763	3.4E-02
g10	7049.33	7054.316	7559.192	8835.655	5.3E+02
g11	0.75	0.75	0.75	0.75	8.0E-05
g12	-1.0	-1.0	-1.0	-1.0	0.0E+00
g13	0.05395	0.053957	0.067543	0.216915	3.1E-02

Table 5. Results obtained by the Stochastic Ranking algorithm [16] performing 350,000 objective function evaluations.

rameters in CLONALG as in Evolution Strategies (ES). In such case, the mutation rate is proportional to the antibodies' affinities using the following equation:

$$\alpha = exp(-\rho f) \tag{5}$$

where α is the step size, ρ controls its decay, and $f(\boldsymbol{x})$ is the antigenic affinity. The sizes of $f(\boldsymbol{x})$ and α are normalized over the interval [0,1].

Based on what we discussed in Section 1, we decided to experiment with both, Gaussian-distributed and Cauchy-distributed random numbers for our mutation operator. The parameter values adopted for our experiments were the following:

- Number of antibodies j: 20
- Percentage d of replaced antibodies: 0.20
- Objective function evaluations: 350,000

The parameters to compute the number of clones using Eq.(4) were the following: $\beta = 1$ and $n = j$.

The parameter to compute α (Eq.(5)) was: $\rho = 20$

6.1 Gaussian-distributed Mutations

In the first set of experiments, the mutation applied to a variable $\boldsymbol{x}_k$ (step 2.d) was computed using: $\boldsymbol{x}_k^{new} = \boldsymbol{x}_k + G(0, \alpha)$, where $G(0, \alpha)$ is a random number between 0 and α with Gaussian distribution.

In order to assess the algorithm's performance after this small modification, we utilized the benchmark described in [16], as before. The results obtained from 30 independent runs using Gaussian-distributed numbers are shown in Table 6. In this case, CLONALG was able to converge to a feasible solution in 11 of the 13 test functions. The quality of the results obtained with Gaussian mutations is similar to that of the algorithm with traditional binary representation. However, the results are outperformed by CLONALG with Gray coding.

Test Function	Optimal	Best	Mean	Worst	Std. Dev.
g01	-15.0	-14.9665	-14.835	-14.559	0.0887
g02	0.803619	-0.7920	-0.710	-0.5	0.0583
g03	1.0	-0.99674	-0.551	-0.0852	0.2943
g04	-30665.5	-30665.2360	-30663.09370	-30661.36	1.0620
g05	5126.4981	INF	INF	INF	INF
g06	-6961.814	-6961.1488	-6949.6321	-6928.9176	8.7312
g07	24.306	31.32150	36.041	40.3094	2.1601
g08	0.095825	-0.095825	-0.095825	-0.095825	0.0000
g09	680.63	682.94	686.4131	689.61	1.7149
g10	7049.33	7099.6346	8953.5650	12146.8123	1698.2081
g11	0.75	1.0	1.0	1.0	0.0000
g12	-1.0	-1.0	-1.0	-1.0	0.0000
g13	0.05395	INF	INF	INF	INF

Table 6. Results obtained by CLONALG with real-numbers representation, using Gaussian mutations. INF means that the algorithm converged to an infeasible solution.

6.2 Cauchy-distributed Mutations

Test Function	Optimal	Best	Mean	Worst	Std. Dev.
g01	-15.0	-14.4501	-13.7009	-12.7895	0.3616
g02	0.803619	-0.35507	-0.3055	-0.256481	0.0264
g03	1.0	0.0	0.0	0.0	0.0
g04	-30665.5	-30664.5824	-30662.4466	-30659.6513	1.3475
g05	5126.4981	INF	INF	INF	INF
g06	-6961.814	-6957.24841	-6917.8961	-6854.0776	26.6042
g07	24.306	40.56171	46.04928	52.1685	3.333
g08	0.095825	-0.095825	-0.09579	-0.0957	0.0
g09	680.63	685.05423	690.51003	696.7897	2.8512
g10	7049.33	7110.1621	8138.4531	11043.2050	986.6993
g11	0.75	1.0	1.0	1.0	0.0
g12	-1.0	-1.0	-1.0	-1.0	0.0
g13	0.05395	INF	INF	INF	INF

Table 7. Results obtained by CLONALG with real-numbers representation, using Cauchy mutations. INF means that the algorithm converged to an infeasible solution.

In our second set of experiments, we tested Cauchy-distributed mutations, aiming to improve the performance of our real-numbers version of CLONALG. In this case, we adopted: $\boldsymbol{x}_k^{new} = \boldsymbol{x}_k + C(0, \alpha)$, where $C(0, \alpha)$ is a random number between 0 and α with Cauchy distribution.

Table 7 shows the results obtained from 30 independent runs using Cauchy-distributed random numbers for our mutation operator. It was quite surprising for us to see that

Cauchy-distributed mutations produced the worst overall results for our real-numbers version of CLONALG. These results seem to indicate that the behavior produced by Cauchy-distributed mutations (emulating a crossover operator) is not the most appropriate when dealing with constrained optimization problems.

7 CLONALG with Controlled and Uniform Mutations

Given the disappointing results that we obtained when using CLONALG in constrained optimization problems, we decided to introduce a modification in its mutation operator aiming to improve the algorithm's performance. Considering that the binary representation version with Gray coding had produced the best results so far, we decided to analyze only the possible changes to the mutation operator for real-numbers representation. Our first modification was to remove the self-adaptation mechanism suggested in [6]. The motivation for this decision was the fact that this self-adaptation mechanism was apparently designed for unconstrained problems and it wasn't obvious to us how to extend it for constrained problems. Thus, it would be easier to analyze the impact of any changes to the mutation operator if this self-adaptation mechanism was removed.

The second change was the introduction of a control mechanism that allowed to increase the algorithm's capability of exploring neighboring regions. This corresponds to step 2.(d) of the algorithm shown in Section 4. All the other steps remained without changes. In this modified mutation operator, the step size is a function of the mutated variable search space size, the antibody's affinity value and the population size.

So, our proposal was to apply mutation in the following way:

1. For each decision variable $\boldsymbol{x}_k$, compute $R_k = UB - LB$, where UB and LB are the upper and lower bounds of that variable, respectively, and R_k is the search space size of the k-th variable.
2. Compute $\Delta_k = R_k/j$ where j is the number of antibodies in the population.
3. The clones population is sorted by affinity values in descending order.
4. The mutation operator is applied to each size-g clone group coming from the same parent.
   (a) For each variable k, compute $\delta_k = \Delta_k/g$.
   (b) Apply mutation to each variable $\boldsymbol{x}_k$ by using $\boldsymbol{x}_k^{new} = \boldsymbol{x}_k + U(0, \delta_k)$, where U is a random number in the range from 0 to δ_k with a uniform distribution.

As the search progresses, the value Δ_k is gradually decreased. The purpose of that is that at the beginning of the search process large mutations are applied to the individuals. Then, as the search progresses (and the algorithm starts converging to a solution), the mutations will become smaller and smaller.

The sorting process mentioned in step 3 is accomplished by placing at the top of the list the antibodies that are feasible and have the best objective function values. After that, we place the infeasible solutions that have the lowest amount of constraint violation and so on. Note how the step sizes of this mutation operator depend on the range of each decision variable, on the size of the antibodies' population and on their affinity.

As mentioned earlier, higher affinity antibodies are allowed to generate more clones g (see step 2.(c) from the algorithm in Section 4). Based on this fact, in step 4.(a) we

obtain smaller step sizes when g is large. We argue that this mutation operator increases the exploratory capabilities of the algorithm.

Test Function	Optimal	Best	Mean	Worst	Std. Dev.
g01	-15.0	-14.9874	-14.7264	-12.9171	0.6070
g02	0.803619	-0.8017	-0.7434	-0.6268	0.0414
g03	1.0	-1.000	-1.000	-1.000	0.0000
g04	-30665.539	-30665.5387	-30665.5386	-30665.5386	0.0000
g05*	5126.498	5126.9990	5436.1278	6111.1714	300.8854
g06	-6961.814	-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	0.75	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	0.37900

Table 8. Results obtained by CLONALG with real-numbers representation and controlled uniform mutation. The asterisk (*) indicates a case in which only 90% of the runs converged to a feasible solution.

Our third set of experiments was performed on the same set of test functions as before. However, in this case, our CLONALG implementation performed 350,000 objective function evaluations. The summary of results (from 30 independent runs) is presented in Table 8. It is clear that the new mutation mechanism produced a remarkable improvement in the results. In this case, the algorithm was able to reach the optimal (or best known) solution in 8 of the 13 test functions adopted. These results are competitive with respect to both ASCHEA and Stochastic Ranking.[1]

Our results seem to suggest that the use of local search (i.e., small step sizes in the mutation operator) has a more significant impact on performance when dealing with constrained search spaces. This contrasts with the case of unconstrained multimodal optimization, in which large step sizes are preferred, to avoid converging to a local optimum [18]. However, other issues such as the most proper balance between feasible and infeasible solutions (i.e., to avoid having only feasible solutions at any time during the search process) remain to be explored (we have adopted a user-defined parameter in our approach, but evidently other alternatives need to be explored). This issue in particular, has been found to have a very significant impact on performance when using evolutionary algorithms for solving constrained optimization problems [9, 16] and therefore its importance.

[1] It is worth remembering that ASCHEA performs a much higher number of objective function evaluations than our approach.

8 Conclusions and Future Work

We have presented a study of the use of the CLONALG approach for solving constrained optimization problems. As part of our study, we have experimented with both binary and real-numbers representation. In the case of binary representation, we also studied the impact of Gray coding, which we found to be positive in terms of the performance of CLONALG.

Regarding real-numbers encoding, we analyzed the use of both Gaussian and Cauchy random numbers. Surprisingly, the use of Cauchy-distributed mutations (which have been found useful in unconstrained numerical optimization) resulted in the worst overall performance of CLONALG.

The poor results obtained in our experiments led us to propose an alternative mutation operator for real-numbers representation. In our proposed mutation operator, the step size depends not only of the antibodies' affinity, but also of the allowable range of each decision variable and of the size of the antibodies' population. This mutation operator was implemented using random numbers with a uniform distribution. As seen in our results, the use of this operator significantly improved the performance of CLONALG with real-numbers representation.

Although there is evidently more room for improvement (we still cannot outperform Stochastic Ranking), the main aim of this paper was to point out the need to do more research on the potential use of CLONALG (and other artificial immune systems) for constrained optimization. As we have seen in this paper, the mechanisms that have been proposed for unconstrained optimization (even if dealing with multimodal functions) are not necessarily the most appropriate for dealing with constrained optimization problems. However, we have also seen that the search capabilities of algorithms such as CLONALG can be regulated through a more carefully designed mutation operator as to provide a competitive performance in constrained optimization. However, other issues such as robustness, balance between feasible and infeasible solutions and how sensitive the algorithm is to the parameters given remain as part of our future work.

9 Acknowledgements

We thank the comments of the anonymous reviewers which greatly helped us to improve the contents of this paper. The first and third authors gratefully acknowledge support from NSF-CONACyT through project 42435-Y. The second author acknowledges support from CONACyT through a scholarship to pursue graduate studies at CINVESTAV-IPN.

References

1. Jerzy Balicki. Multi-criterion evolutionary algorithm with model of the immune system to handle constraints for task assignments. In L. Rutkowski, J. Siekmann, R. Tadeusiewicz, and L.A. Zadeh, editors, *Artificial Intelligence and Soft Computing – ICAISC 2004 7th International Conference, Proceedings*, volume 3070, pages 394–399. Springer, Lecture Notes in Computer Science, 2004.

2. 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.
3. David Corne, Marco Dorigo, and Fred Glover, editors. *New Ideas in Optimization*. McGraw-Hill, London, UK, 1999.
4. Leandro Nunes de Castro and Jon Timmis. An artificial immune network for multimodal function optimization. In *Proceedings of the special sessions on artificial immune systems in the 2002 Congress on Evolutionary Computation, 2002 IEEE World Congress on Computational Intelligence*, volume I, pages 669–674, Honolulu, Hawaii, May 2002.
5. Leandro Nunes de Castro and Jonathan Timmis. *An Introduction to Artificial Immune Systems: A New Computational Intelligence Paradigm*. Springer-Verlag, 2002.
6. Leandro 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.
7. Raziyeh Farmani and Jonathan A. Wright. Self-Adaptive Fitness Formulation for Constrained Optimization. *IEEE Transactions on Evolutionary Computation*, 7(5):445–455, October 2003.
8. Prabhat Hajela and Jun Sun Yoo. Immune network modelling in design optimization. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 167–183. Mc Graw-Hill, 1999.
9. S. B. Hamida and M. Schoenauer. ASCHEA: New results using adaptive segregationsl constraint handling. In *Proceedings of the Congress on Evolutionary Computation 2002 (CEC'02)*, volume 1, pages 884–889, Piscataway, New Jersey, 2002. IEEE Service Center.
10. J. Kelsey and J. Timmis. Immune Inspired Somatic Contiguous Hypermutation for Function Optimisation. In E. Cantú-Paz et al, editor, *Genetic and Evolutionary Computation Conference - GECCO 2003 of Lecture Notes in Computer Science*, volume 2723, pages 207–218, Chicago, USA., 2003. Springer-Verlag.
11. Slawomir Koziel and Zbigniew Michalewicz. Evolutionary Algorithms, Homomorphous Mappings, and Constrained Parameter Optimization. *Evolutionary Computation*, 7(1):19–44, 1999.
12. G. C. Luh and C. H. Chueh. Multi-objective optimal designof truss structure with immune algorithm. *Computers and Structures*, 82:829–844, 2004.
13. G. C. Luh, C. H. Chueh, and W. W. Liu. MOIA: Multi-Objective Immune Algorithm. *Engeneering Optimization*, 35(2):143–164, 2003.
14. Keith E. Mathias and L. Darrel Whitley. Transforming the search space with Gray coding. In J. D. Schaffer, editor, *Proceedings of the IEEE International Conference on Evolutionary Computation*, pages 513–518. IEEE Service Center, Piscataway, New Jersey, 1994.
15. Zbigniew Michalewicz and Marc Schoenauer. Evolutionary Algorithms for Constrained Parameter Optimization Problems. *Evolutionary Computation*, 4(1):1–32, 1996.
16. Thomas P. Runarsson and Xin Yao. Stochastic ranking for constrained evolutionary optimization. *IEEE Transactions on Evolutionay Computation*, 4(3):284–294, 2000.
17. 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.
18. X. Yao and Y. Liu. Fast evolution strategies. *Control and Cybernetics*, 26(3):467–496, 1997.
19. J. Yoo and P. Hajela. Enhanced GA Based Search Through Immune System Modeling. In *3rd. World Congress on Structural and Multidisciplinary Optimization*. IEEE Press, 1999.