Herrera-Lozada, Juan Carlos; Calvo, Hiram; Taud, Hind
A Micro Artificial Immune System
Polidbits, vol. 43, 2011
Instituto Politécnico Nacional
Distrito Federal, México

Available in: http://www.redalyc.org/articulo.oa?id=402640456015
Abstract—In this paper, we present a new algorithm, namely, a micro artificial immune system (Micro-AIS) based on the Clonal Selection Theory for solving numerical optimization problems. For our study, we consider the algorithm CLONALG, a widely used artificial immune system. During the process of cloning, CLONALG greatly increases the size of its population. We propose a version with reduced population. Our hypothesis is that reducing the number of individuals in a population will decrease the number of evaluations of the objective function, increasing the speed of convergence and reducing the use of data memory. Our proposal uses a population of 5 individuals (antibodies), from which only 15 clones are obtained. In the maturation stage of the clones, two simple and fast mutation operators are used in a nominal convergence that works together with a reinitialization process to preserve the diversity. To validate our algorithm, we use a set of test functions taken from the specialized literature to compare our approach with the standard version of CLONALG. The same method can be applied in many other problems, for example, in text processing.

Index terms—Artificial immune system, Clonal selection theory, micro algorithm, numerical optimization.

I. INTRODUCTION

Bio-inspired and Evolutionary algorithms have become very important within the area of artificial intelligence, because they have proved successful in solving certain complex problems of machine learning, classification and numerical optimization [1]. Such techniques are population based, in other words, they use a population of potential solutions enabling a wide exploration of the search space. The simplicity of an algorithm is one of the current trends in the field of evolutionary computation, although in the majority of cases, the performance is sacrificed in favor of a lower computational cost [2]. Due to the simultaneous manipulation of a large set of solutions, implementation of an algorithm requires a large space in data memory and generally high processing time. To reduce these factors, algorithms with extremely small populations are designed, and for most applications their performance is comparable with the standard population algorithms [3].

As the evolutionary algorithms, Artificial Immune System (AIS) has been successfully applied to a variety of optimization problems [4]. AIS is a computational intelligence paradigm inspired by the biological immune system which has found application in pattern recognition and machine learning. Different ways of AIS for optimization as the immune network theory and the clonal selection principle have been proposed and implemented by different researchers as explained in [5].

The main motivation of our research is to propose a simple and powerful algorithm which presents a reduced computational cost when using a micro population of individuals within a clonal proliferation scheme which is the central point in the functioning of artificial immune system.

Two novel mutation operators were designed and implemented. These operators accelerate the convergence by providing a uniform search to avoid getting into local optimum.

In this work we apply micro-AIS for numerical optimization, but the same method can be applied in many other problems, for example, in text processing. It is promising to apply bio-inspired algorithms (more specifically, genetic algorithms) in text processing tasks, see, for example, [6].

A. Previous Work

In [3] Goldberg introduced the concept of nominal convergence when he experimented with a simple genetic algorithm (GA) using a population of only 3 individuals. He found that these 3 chromosomes were sufficient to ensure convergence of the algorithm regardless of the size of them, aided by a process of elitism. Goldberg applied genetic operators in a nominal convergence which is controlled by two possible parameters: a specified number of generations or a degree of similarity among all chromosomes. At the end of the nominal convergence, the best individual is preserved and two individuals are randomly generated: they will form the new population.

In [7] Krishnakumar designed a GA with a population of 5 individuals and he named his algorithm Micro Genetic Algorithm (Micro-GA). Like Goldberg, Krishnakumar used elitism to preserve the best single strand found at the end of nominal convergence, as one of the individuals used for the next generation. When comparing the performance of the Micro-GA with a simple GA with a population of 50 individuals, better results were obtained on functions of only one objective and the GA with a reduced population converged faster. Krishnakumar’s algorithm has achieved good results when it is used to solve optimization problems for high-dimensional functions [8].

Dozier et al. in [9] presented two heuristic-based micro genetic algorithms which quickly find solutions to constraints satisfaction problem. They experimented with different sizes of micro population and found that for a particular problem, a relatively small number of individuals in the genetic algorithm was sufficient.
Coello and Toscano designed a Micro-GA for solving the multi-objective optimization problem [10], providing criteria for the management of constraints, besides proposing a scheme of Pareto dominance with a geographical location to maintain the diversity and uniformly distribution of the solutions on the Pareto front. This algorithm works with a population of 4 individuals and uses a secondary memory that stores potential solutions throughout the search. This approach was widely used to successfully solve various engineering problems as discussed in [11] and [12].

Recently, Fuentes and Coello in [13] designed a micro algorithm for PSO (Particle Swarm Optimization) to solve optimization problems of one objective and constraints satisfaction. They use 5 particles (individuals) helped by a nominal convergence.

With regard to artificial immune systems with small population, there are no studies reported in the literature. There are, however, certain similarities with the works cited above:

1. Population size of 3 to 5 individuals.
2. Nominal convergence is required as well as a reinitialization process.
3. Elitism is necessary to preserve at least the best individual obtained at the end of the nominal convergence.

II. ARTIFICIAL IMMUNE SYSTEM

De Castro and Von Zuben developed the Clonal Selection Algorithm (CLONALG) on the basis of clonal selection theory of the immune system [14, 15]. 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. This algorithm can perform pattern recognition and adapt to solve multimodal optimization tasks. The block diagram of CLONALG is shown in Fig. 1. This algorithm is described as follows:

1. Generate (randomly) a set \( P \) of candidate solutions or antibodies, composed of the memory cells \( M \) and the remaining population \( P_r \), \( \left( P = P_r + M \right) \);
2. Select the \( n \) best antibodies \( P_n \), based on affinity;
3. Clone these \( n \) best antibodies in proportion to their affinity using \( N_c = \sum_{i=1}^{n} \left( \frac{P_i}{\sum P} \right) \) where \( N_c \) is the total number of clones generated for each of the antigens like objective function, \( \beta \) is a multiplying factor, \( N \) is the total number of antibodies, and \( \lfloor \cdot \rfloor \) is the operator that rounds its argument toward the closest integer. Each term of this sum corresponds to the clone size of each selected antibody, e.g., for \( N=100 \) and \( \beta = 1 \), the antibody with highest affinity will produce 100 clones; the antibody with the second highest affinity produces 50 clones, and so on, giving rise to a temporary set of clones \( C \);
4. Apply a hypermutation to the temporary clones. The degree of mutation is inversely proportional to the affinity. The maturated antibodies are generated \( C^* \);
5. Re-select the best elements from \( C^* \) to compose the memory set \( M \). Some members of \( P \) can be replaced by other improved members of \( C^* \);
6. Replace \( d \) antibodies by novel ones to introduce the diversity concept. The probability to be replaced is inversely proportional to the affinity of the previous remaining population \( P_r \).

Fig. 1. Block diagram of the clonal selection algorithm CLONALG by De Castro and Von Zuben.

Fig. 2. Micro-AIS.
III. MICRO ARTIFICIAL IMMUNE SYSTEM

Fig. 2 shows our algorithm. Our methodology is based on the methodology proposed by Goldberg in [3]: the variation operators are applied to a small population (randomly generated) to achieve nominal convergence. Subsequently, a new population should be generated by transferring the best individuals of the population obtained after the convergence to the new one. The remaining individuals are randomly generated.

The proposed algorithm works as follows:

1. Generate randomly a population of 5 antibodies (individuals). In the initial generation, these antibodies are copied directly to the working population and nominal convergence is controlled by the number of generations, in our case equal to 10.

2. Use selection based on ranking. The antibody with the highest affinity will be the best individual. In our algorithm we named this individual as BestAb.

3. Perform the cloning of the antibodies using $N_C = \sum_{i=1}^{n} (n - (i - 1))$, where $N_C$ is the number of clones to be generated for each antibody, $n$ is the total number of antibodies of the population and $i$ is the current antibody starting from the antibody with the highest affinity (BestAb).

4. Consider a population of 5 antibodies and generate a population of 15 clones: BestAb antibody gets 5 clones; the second ranking antibody gets 4 clones and so on until the worst antibody that gets a single clone.

5. Perform the maturation of clones using mutation process. The probability of mutation is set at the beginning of nominal convergence for each group of clones obtained from the same antibody. This probability is determined in proportion to the affinity of each antibody and decreases uniformly in each generation, so the group of clones obtained from BestAb mutates less than other groups of clones that have been generated from the remaining antibodies. The single clone that we got from the worst antibody has the highest possibility to mutate. For this purpose we use

$$\text{prob\_mutation}(i) = \frac{\text{Aff}(i)}{\sum_{i=1}^{n} \text{Aff}(i)}$$

where $i$ is the antibody that will set the mutation probability for the group of clones that were obtained from himself and $n$ is the total antibody population. To decrease the mutation probability uniformly in each generation, within the nominal convergence we used

$$\text{prob\_mutation}(i) = \frac{\text{Aff}(i)}{\sum_{i=1}^{n} \text{Aff}(i)}$$

where $i$ is the antibody that will set the mutation probability for the group of clones that were obtained from himself and $n$ is the total antibody population. To decrease the mutation probability uniformly in each generation, within the nominal convergence we used

$$\text{prob\_mutation}(i) = \frac{\text{Aff}(i)}{\sum_{i=1}^{n} \text{Aff}(i)}$$

and

$$x' = x + \left( \frac{a \cdot \text{range} \cdot \text{generation}}{N_C} \right)$$

where $x'$ is the mutated decision variable, $x$ is the decision variable to mutate, $a$ is a uniform random number where $\text{random } a \in [0,1]$, $\text{generation}$ is the current generation within the nominal convergence and $N_C$ is the total number of clones.

6. Make another selection based on ranking. This time, we sort the 15 clones with respect to their affinity. We must select the two best clones (elitism) and the new population is completed with 3 other clones selected randomly from the population of mature clones. The remaining clones will be eliminated, providing a self-regulation within the nominal convergence.

7. When nominal convergence is achieved (while working with 10 generations), we keep the two best clones, and other 3 antibodies are generated randomly to complete the new working population and the nominal convergence starts again until the algorithm achieves the stop condition.

IV. EXPERIMENTAL SETUP

In order to validate the proposed approach, we used the multivariate functions presented in [16]. These functions are listed in Appendix A. All selected test functions have 30 variables (dimensions) and an optimum value at zero, except for $f08$ with an optimum at -12569.5. For all cases we used a population of 5 individuals and nominal convergence in 10 generations. The general stop criterion of the algorithm varied depending on the problem to be solved. For the experiments, we used a 2.66 GHz Quad Core PC with 2MB. Table I shows the results for 20 runs of the algorithm.
To validate the performance of our algorithm with respect to the standard version of CLONALG, we compared it with some of the above mentioned functions under equal conditions. The main results are related with the number of evaluations of the objective function and convergence time. Table II lists these results for 20 runs of both algorithms. We implemented the adaptations to CLONALG for using multivariate functions. For CLONALG we used a multiplication factor $\beta = 1$ and the number of antibodies listed in Table II.

### V. CONCLUSIONS AND FUTURE WORK

In this paper we presented a new micro algorithm based on clonal selection theory for solving numerical optimization problem. Since the model of the artificial immune system does not include a crossover operator, cloning (set to 15 clones in our case) and mutation represent the main challenges for maintaining diversity.

Two mutation operators were designed in our approach showed excellent solutions with a low computational cost. These operators were used without modifications in all selected test functions. As shown in the results listed in Tables I and II, the Micro-AIS converges faster than CLONALG and uses less data memory. The nominal convergence and elitism of 40% of the population (considering only 5 antibodies) are of great importance to ensure the proper functioning of the algorithm.

Future work is aimed at the following four aspects:
- Design versions for handling constraints and multi-objective optimization,
- Develop possible hardware architectures, and
- Develop applications in different areas (for example, in text processing) and experiment with them.

### REFERENCES


APPENDIX A

Multivariate functions for the experimental setup, taken from [15].

**f01** – Sphere Model

\[ f_1(x) = \sum_{i=1}^{30} (x_i)^2 \]

-100 ≤ \( x_i \) ≤ 100
\[ \min \{f_1\} = f_1(0, \ldots, 0) = 0 \]

**f02** – Schwefel’s Problem

\[ f_2(x) = \sum_{i=4}^{30} |x_i| + \prod_{i=1}^{30} |x_i| \]

-10 ≤ \( x_i \) ≤ 10
\[ \min \{f_2\} = f_2(0, \ldots, 0) = 0 \]

**f03** – Schwefel’s Problem

\[ f_3(x) = \sum_{i=1}^{30} \left( \sum_{j=1}^{i} x_j \right)^2 \]

-100 ≤ \( x_i \) ≤ 100
\[ \min \{f_3\} = f_3(0, \ldots, 0) = 0 \]

**f04** – Schwefel’s Problem

\[ f_4(x) = \max_i \{ x_i \} \quad 1 ≤ i ≤ 30 \]

-100 ≤ \( x_i \) ≤ 100
\[ \min \{f_4\} = f_4(0, \ldots, 0) = 0 \]

**f05** – Generalized Rosenbrock’s Function

\[ f_5(x) = \sum_{i=1}^{30} \left[ 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2 \right] \]

\(-30 ≤ x_i ≤ 30 \quad \min \{f_5\} = f_5(1, \ldots, 1) = 0 \)

**f06** – Step Function

\[ f_6(x) = \sum_{i=1}^{30} \left( x_i + 0.5 \right) \]

-100 ≤ \( x_i \) ≤ 100
\[ \min \{f_6\} = f_6(0, \ldots, 0) = 0 \]

**f07** – Quartic Function with Noise

\[ f_7(x) = \sum_{i=1}^{30} x_i^4 + \text{random}[0,1] \]

-1.28 ≤ \( x_i \) ≤ 1.28
\[ \min \{f_7\} = f_7(0, \ldots, 0) = 0 \]

**f08** – Generalized Schwefel’s Problem

\[ f_8(x) = \sum_{i=1}^{30} x_i \sin \left( \frac{\sqrt{\sum_{i=1}^{30} x_i}}{\sqrt{30}} \right) \]

-500 ≤ \( x_i \) ≤ 500
\[ \min \{f_8\} = f_8(420.9687, \ldots, 420.9687) = -12596.5 \]

**f09** – Generalized Rastrigin’s Problem

\[ f_9(x) = \sum_{i=1}^{30} \left( x_i^2 - 10 \cos (2\pi x_i) + 10 \right) \]

-5.12 ≤ \( x_i \) ≤ 5.12
\[ \min \{f_9\} = f_9(0, \ldots, 0) = 0 \]

**f10** – Ackley’s Function

\[ f_{10}(x) = -20 \exp \left(-0.2 \sqrt{\frac{1}{30} \sum_{i=1}^{30} x_i^2} \right) - \exp \left( \frac{1}{30} \sum_{i=1}^{30} \cos (2\pi x_i) \right) + 20 + e \]

-32 ≤ \( x_i \) ≤ 32
\[ \min \{f_{10}\} = f_{10}(0, \ldots, 0) = 0 \]

**f11** – Generalized Griewank’s Function

\[ f_{11}(x) = \frac{1}{4000} \sum_{i=1}^{30} x_i^2 - \prod_{i=1}^{30} \cos \left( \frac{x_i}{\sqrt{i}} \right) + 1 \]

-600 ≤ \( x_i \) ≤ 600
\[ \min \{f_{11}\} = f_{11}(0, \ldots, 0) = 0 \]