XLI CILAMCE
XLI Ibero-Latin American Congress on Computational Methods in Engineering
16-19 November, 2020 FOZ DO IGUAÇU PARANÁ - BRAZIL

Comparative study of the performance of different bio-inspired algorithms using the same cost function

Pedro Henrique Nunes[1]

[1]*Automatic Department, University Federal of Lavras*
Praça Prof. Edmir Sá Santos, S/N, 37200-900, Lavras, Minas Gerais, Brazil
pedro.nunes@estudante.ufla.br

Abstract. Evolutionary computing is a computational intelligence tool widely used for data analysis and cost functions. In addition to being accessible in different languages, there are different types of techniques and concepts that can be used for different types of functions. One of these techniques is the bio-inspired algorithms, which have this name due to the fact that they are inspired by phenomena occurring in nature to search for results. One of the situations where this type of algorithm is generally used is in minimizing or maximizing a value of a cost function. There are several types and models of bio-inspired algorithms in the literature, each inspired by a phenomenon and which works best on a certain type of problem. This work aims to explore different types of these algorithms and thus perform a comparative study of which has the best performance for a function-cost minimization problem, here represented by the function called Matyas, which has two dimensions and has no local minimums , only the global. For this, 7 different types of algorithms were designed, which are: Simple Differential Evolution, Differential Evolution with Variant, Differential Evolution with Opposition, Simple PSO, PSO with Constriction Factor, PSO with Inertia Factor and CLONALG. The results showed that, for the chosen optimization cost-function, as the objective was to minimize the value of the variable, the Differential Evolution algorithms obtained more satisfactory results in relation to the PSO and CLONALG algorithms, especially with opposition learning.

Keywords: Nonlinear parameterization; Genetic Algorithm; Hybrid method.

1 Introduction

Evolutionary computing is a computational intelligence tool widely used in different problem solutions in terms of data analysis and cost functions. In addition to being accessible in different languages, it is possible to see several other advantages, such as the different types of techniques and concepts that can be used for different types of functions.

In this sense, one of the most used techniques in computational intelligence is the bio-inspired algorithms, which have this name due to the fact that they are inspired by phenomena occurring in nature to search for results. Thus, such algorithms can be used in different types of problems, one of the situations where it is generally used is in minimizing or maximizing a value of a cost function.

There are several types and models of bio-inspired algorithms in the literature, each inspired by a phenomenon and which works best on a certain type of problem. This work aims to explore different types of these algorithms and thus perform a comparative study of which has the best performance for a function-cost minimization problem, here represented by the function called Matyas, which has two dimensions and has no local minimums , only the global.

For this, 7 different types of algorithms were designed, which are: Simple Differential Evolution, Differential Evolution with Variant, Differential Evolution with Opposition, Simple PSO, PSO with Constriction Factor, PSO with Inertia Factor and CLONALG. The results showed that, for the chosen optimization cost-function, as the objective was to minimize the value of the variable, the Differential Evolution algorithms obtained more satisfactory results in relation to the PSO and CLONALG algorithms, especially with opposition learning.

This article is organized into sections. Section II will provide an overview of the cost function used and how the algorithms used work. Section III presents the results, which will be discussed in section IV, where the conclusions of this work will also be presented.

2 Materials and Methods

2.1 Cost Function

The cost function chosen for this project consists of an optimization function and seeks to find the minimum location in a characteristics plan. It is called the Matyas Function, due to the name of its creator. Its graph, together with its formula, can be seen in Figure 1 and Equation 1.

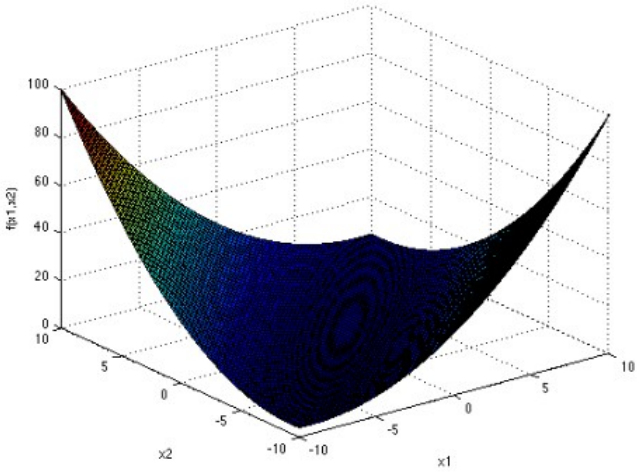

Figure 1. Matyas Function Graph.

$$f(x) = 0.26(x_1^2 + x_2^2) - 0.48x_1x_2 \tag{1}$$

2.2 PSO

Developed by social psychologist James Kennedy and electrical engineer Russel Eberhart in 1995 [1], PSO is an optimization technique that emerged from the analysis of the social behavior of flocks of birds. In the algorithm each individual is called a particle and behaves like a flock bird looking for food or the location of its nest. To reach its objective, the particle uses the learning acquired by its own experiences and also the learning of the flock (or swarm). Because it has an emergent behavior, in a cloud of particles there is no a central control. Each particle acts and makes decisions based on local information and global, as in other Artificial Life techniques [2].

Thus, evaluation, comparison and imitation are important properties of human social behavior and, therefore, are the basis for the particle cloud, which uses these concepts in adapting to changes in the environment and in solving complex problems [1].

PSO is considered a multi-agent system to solve complex optimization problems. A striking feature of this algorithm is the way in which these agents, or particles, are related to each other. They exhibit collaborative behavior, simulating the sharing of experiences and cooperating with each other in search for the best solutions. Like PSO, another SI algorithm also has this feature: Ant Colony Optimization - Ant Colony Optimization (ACO), used as a comparison with the algorithm of this work in chapter 8.1. But not all metaheuristic algorithms have this characteristic in the relationship among population agents. In the case of evolutionary computing algorithms, for example, the governing behavior is just the opposite, the agents compete among themselves to perpetuate their own characteristics for the next iteration, or generation, thus simulating Darwin's theory of the evolution of species (1872).

In the PSO algorithm the possible solutions, called particles, travel through space of the problem, following the best positions found so far by the particles of the flock. These best positions, which the particles seek to follow, can be classified into three distinct groups: the best position found by herself until the moment, called best position found by the particles neighboring it, called the best position found by the entire population taking into account all particles, called sition of best overall value. As in all optimization problem based on populations, at the end of the execution the best or the best solutions, according to an objective function, are presented as a result.

As defined by [3] in the PSO each particle represents a possible solution and is represented as a position in the state space. The idea of PSO is to execute a set of operators and move each particle to regions promising in the

search space. At each iteration the particles are changed as if moved in a n-dimensional space and, thus, a new set of solutions is obtained. This occurs by applying their respective speeds. Every iteration the speed of each particle is adjusted. The speed calculation is based on the best position found by the particle's neighborhood and the best position found by the particle itself.

2.3 CLONALG

The clonal selection algorithm (CLONALG - Clonal Selection Algorithm) is inspired by principle of clonal selection that occurs in the biological immune system. This algorithm was proposed by [4]. Following is the CLONALG algorithm applied in this work:

- Step 1: Generate a population (P) with N antibodies (candidate solutions);
- Step 2: Assess the affinity (objective function) of each antibody and select (selection process) the n best antibodies of the P population, obtaining the set P n;
- Step 3: Reproduce (cloning process) the numbers best selected antibodies, generating a population (C) with Nc clones. The number of clones of each antibody is directly proportional to its affinity;
- Step 4: Submit the population of clones (C) to a hypermutation process, where the rate of mutation is inversely proportional to the affinity of the antibody. A population (C *) of antibodies mature and matured is generated;
- Step 5: Assess the affinity of each antibody belonging to (C *) and re-select the best n antibodies (C *n) and replace them with the P population.
- Step 6: Replace low affinity antibodies by new antibodies (P d) (diversity or metadinamics). Antibodies with low affinity are more likely to be replaced;
- Step 7: Repeat steps 2 through 6 until you meet the stopping criterion.

Antibodies (proposed solutions) can be encoded in real or binary format according to the problem. Each antibody generates an amount total (Nc) of clones. Clones can suffer mutations at a rate inversely proportional to affinity (objective function). During the execution of the algorithm, antibodies with less affinity (diversity) are replaced by new antibodies, randomly generated. The Nc quantity of clones generated in Step 3 to each antibody i is given by Equation 2[4]:

$$N_c{}^i = round\left(\frac{\beta N}{i}\right) \tag{2}$$

where: β is a multiplicative factor between [0.1], N is the total amount of antibodies in the P population, and round (.) is the rounding operator for the nearest integer.The mutation rate (α) of each clone is defined by the Equation 3 [4]:

$$\alpha = exp(-\rho D^*) \tag{3}$$

where: ρ is a control parameter of smoothing the exponential function, D * is the value normalized affinity D, Dmax is the highest value of affinity and Dmin is the lowest affinity value. Can be calculated as presented in Equation 4 for maximization problems and Equation 5 for minimization problems.

$$D^* = \frac{D}{D_{max}} \tag{4}$$

$$D^* = \frac{D_{min}}{D} \tag{5}$$

In this way, each clone undergoes a process of mutation given by [5]:

$$m = round(\alpha * N(0,1)) \tag{6}$$

where: m the number of mutations that each clone of the antibody will suffer, round (.) is the operator of rounding to the nearest integer, α is the mutation rate and N (0.1) is a random variable Gaussian mean zero and standard deviation σ = 1.

2.4 Differential evolution

The Differential Evolution (DE) algorithm is a simple and efficient optimization algorithm that was proposed by Rainer Storn and Kenneth Price in 1995 [6]. It is a stochastic search method that initially appeared in order to solve a problem of Chebychev polynomial adjustment.

DE is a method, which besides being easy to implement, performs very well in a large class of problems, as reported by [6]. It is effective for objective functions that are not differentiable or convex and it is easy to find the optimum with small populations.

The Differential Evolution algorithm has few control variables, whose adjustment is simple. It has good convergence properties with an auto adjustment of the adaptation step, that is, as the convergence the steps are smaller.

There are three types of Differential Evolution algorithms, which are: Basic, Variant and Opposition. The Basic DE algorithm for each vector tag $x_i, G, i = 1$ a new vector is generated using the Equation 7.

$$x_{i,G+1} = x_{r_1,G} + F * (x_{r_3,G} - x_{r_2,G}) \tag{7}$$

where r_1, r_2 and r_3 are indexes that are mutually distinct and also different from index i, and F is a constant that determines the size of the step to be taken.

There are still DE algorithms for Variants, which consist of changing the attributes of the cost function; and also by opposition, where a candidate solution and its opposite must be found at the same time.

In this work, the DE algorithm was submitted to a problem that minimizes the residual standard error of the estimate, where the individuals were composed of the value of β_0 and β_1. To calculate the residual standard error, the parameter values were applied to the model (Equation 7) obtaining the estimate and then the error in each observation.

2.5 TLBO

The TLBO algorithm is an algorithm inspired by the teaching-learning process and is based on the effect of a teacher's influence on the production of students in a class. The algorithm describes two basic modes of learning: $a)$ through the teacher known as the teacher's phase; $b)$ through interaction with other students (known as the student phase). In this optimization algorithm, a group of students is considered a population and different subjects offered to students are considered different design variables from the optimization problem and the result of a student is analogous to the 'adequacy' value of the optimization problem.

The best solution in the entire population is considered to be a teacher. The design variables are actually the parameters involved in the objective function of the optimization problem provided and the best solution is the best value of the objective function.

Like DE, the TLBO algorithm was subjected to a problem that minimizes the residual standard error of the estimate, where individuals were composed of the value of β_0 and β_1. To calculate the residual standard error, the parameter values were applied to the model (Equation **??**) obtaining the estimate and then the error in each observation.

3 Results

In this section, the results obtained with the implementation of the algorithms will be presented. The Differential Evolution algorithm was developed taking into account its three variants, which are: the pure algorithm, the algorithm plus the variant and the algorithm plus the opposition. Similarly, the algorithm was applied taking into account the simple algorithm, the algorithm plus the inertia factor and the algorithm with the constriction factor.

To evaluate this work, an experiment was carried out with 10 repetitions of each method, where the local minimum values found by each type of algorithm developed were compared.

The results will be presented in graphical and tabular form. In the graphs, points and lines represent the minimum values obtained by the performance of each algorithm. Figures 6, 7 and 8 show the results obtained with the PSO algorithm and its variations.

In Figure 5, the result obtained with the CLONALG algorithm is shown.

Figures 6, 7 and 8 show the results obtained with the Differential Evolution algorithm and its variations.

In Figure 9, the result obtained with the TLBO algorithm is shown.

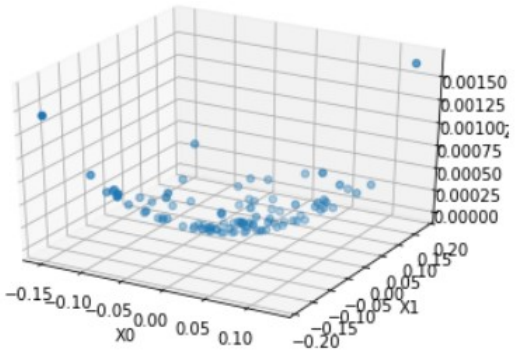

Figure 2. Particle behavior in the simple PSO algorithm.

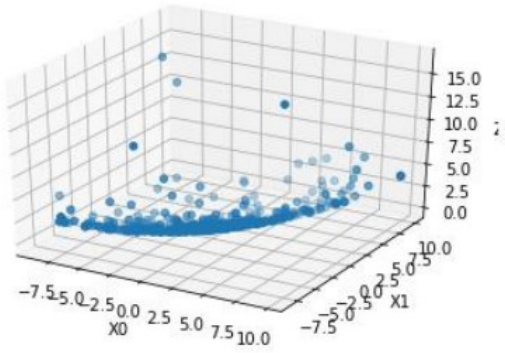

Figure 3. Particle behavior in the PSO algorithm with constriction factor.

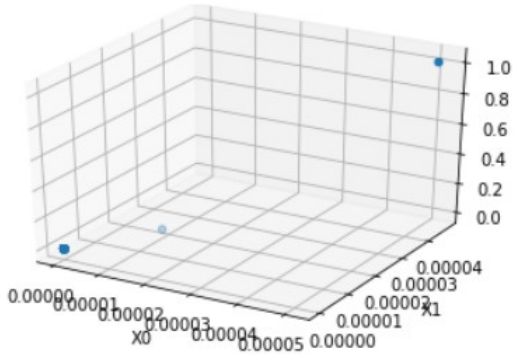

Figure 4. Particle behavior in the PSO algorithm with inertia factor.

4 Conclusions

Table 1 shows the best quantitative results obtained in each of the implemented algorithms.

Note that, for the chosen optimization cost-function, as the objective was to minimize the value of the variable, the Differential Evolution algorithms obtained more satisfactory results in the PSO, CLONALG and TLBO algorithms.

From the computational cost point of view, the PSO and Differential Evolution algorithms can result in higher costs. Therefore, despite obtaining the best performance, the problem itself is who will determine the best algorithm to be used.

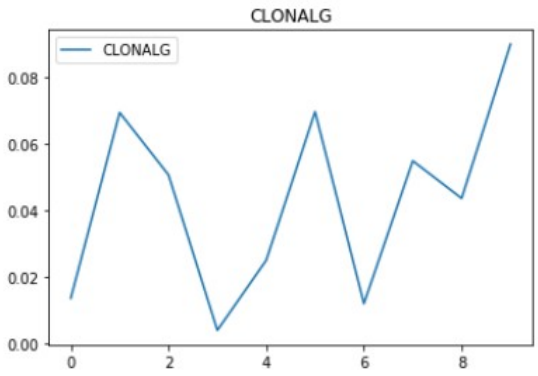

Figure 5. Data behavior during generations of the CLONALG algorithm.

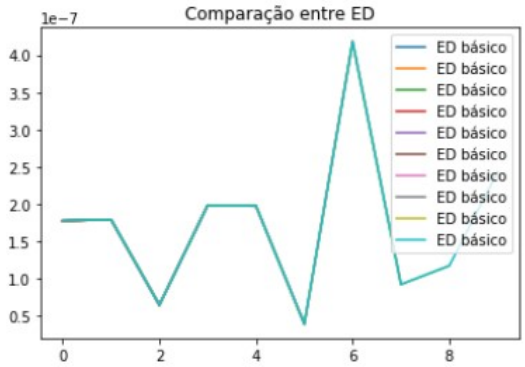

Figure 6. Particle behavior in the Simple Differential Evolution algorithm.

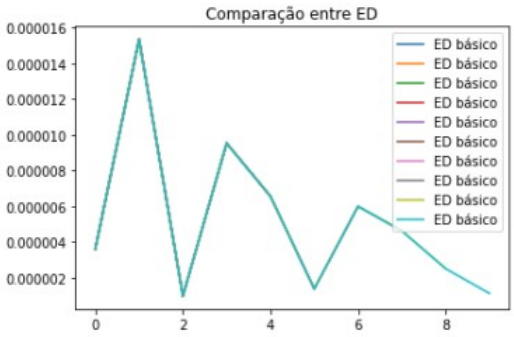

Figure 7. Particle behavior in the Differential Evolution algorithm with Variant.

Acknowledgements. The author would like to thank UFLA, its faculty, management, administration and all the people who have lived in these spaces over the years and for the opportunity to show the work to society.

Authorship statement. The authors hereby confirm that they are the sole liable persons responsible for the authorship of this work, and that all material that has been herein included as part of the present paper is either the property (and authorship) of the authors, or has the permission of the owners to be included here.

References

[1] KENNEDY, J.; EBERHART, R. C., 1995. Particle swarm optimization. *Proceedings of IEEE International Conference on Neural Networks.*, pp. 1942–1948.

[2] VESTERSTRØM, J. S.; RIGET, J., 2002. Particle swarms: Extensions for improved local, multi-modal, dynamic search in numerical optimization. *Faculty of Science, Aarhus Universitet.*

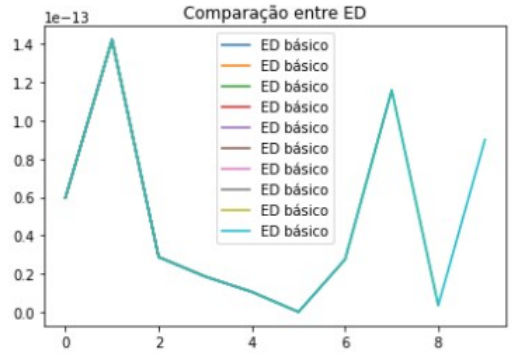

Figure 8. Particle behavior in the Differential Evolution algorithm with Opposition.

Figure 9. Data behavior during generations of the TLBO algorithm.

Table 1. Results obtained after processing evolutionary algorithms.

Algorithm	Best value
Simple PSO	$3.355626140e^{5}$
PSO with constriction factor	0.000774234
PSO with inertia factor	$3.04987496e^{-14}$
CLONALG	0.000654
Simple Differential Evolution	$8.57934e^{-14}$
Differential Evolution with Variant	$4.27475e^{-11}$
Differential Evolution with Opposition	$8.80247e^{-15}$
TLBO	0.0461

[3] CARVALHO, A. B. d., 2008. Otimização por nuvem de partículas multiobjetivo no aprendizado indutivo de regras: extensões e aplicacações. dissertação (mestrado). *Universidade Federal do Paraná.*
[4] de Castro, L. N. & Zuben., J. F. V., 2000. The clonal selection algorithm with engineering applications. in: Workshop proceedings of gecco, workshop on artificial immune systems and their applications. pp. 36–39.
[5] de Franca, F. O., V. Z. F. J. d. C. L. N., 2005. An artificial immune network for multimodal function optimization on dynamic environments. *Proc. GECCO, Washington.*
[6] Moré, J. J., 1978. The levenberg-marquardt algorithm: Implementation and theory. In Watson, G. A., ed, *Numerical Analysis*, pp. 105–116, Berlin, Heidelberg. Springer Berlin Heidelberg.