# Extremal Dynamics Applied to Function Optimization

Fabiano Luis de Sousa[*,1], Fernando Manuel Ramos[**,1]

(1) Área de Computação Científica
Laboratório Associado de Computação e Matemática Aplicada
Instituto Nacional de Pesquisas Espaciais (INPE)

(*) Doutorado, email: fabiano@dem.inpe.br; (**) Orientador

**Abstract**

In this article is introduced the Generalized Extremal Optimization algorithm. Based on the dynamics of self-organized criticality, it is intended to be used in complex inverse design problems, where traditional gradient based optimization methods are not efficient. Preliminary results from a set of test functions show that this algorithm can be competitive to other stochastic methods such as the genetic algorithms.

Key words: Extremal dynamics, optimization, self-organized criticality.

## Introduction

Recently, Boettcher and Percus [1] have proposed a new optimization algorithm based on a simplified model of natural selection developed by Bak and Sneppen [2]. Evolution in this model is driven by a extremal process that shows characteristics of self-organized criticality (SOC). Boettcher and Percus [1] have adapted the evolutionary model of Bak and Sneppel [2] to tackle hard problems in combinatorial optimization, calling their algorithm Extremal Optimization (EO).

Although algorithms such as Simulated Annealing (SA), Genetic Algorithms (GAs) and the EO are inspired by natural processes, their practical implementation to optimization problems shares a common feature: the search for the optimal is done through a stochastic process that is "guided" by the setting of adjustable parameters. Since the proper setting of these parameters are very important to the performance of the algorithms, it is highly desirable that they have few of such parameters, so that the cost of finding the best set to a given optimization problem does not become a costly task in itself. The EO algorithm has only one adjustable parameter. This may be an "a priori" advantage over the SA and GA algorithms, since they use more than one adjustable parameter.

In the next section we introduce the Generalized Extremal Optimization (GEO) algorithm. It was designed to tackle function optimization problems with multiple local optima. In this work the performance of the GEO algorithm is tested in a set of multimodal unconstrained functions, with side constraints on the design variables, used commonly to test GAs. The GEO algorithm is compared with a standard GA and the Cooperative Co-evolutionary GA (CCGA) proposed by Potter and De Jong [3].

## The Extremal Optimization algorithm applied to function optimization

Self-organized criticality has been used to explain the behavior of complex systems in such different areas as geology, economy and biology [4]. The theory of SOC states that composite systems evolves naturally to a critical state where a single change in one of its elements generates "avalanches" that can reach any number of elements on the system. An optimization heuristic based on a dynamic search that embodies SOC could evolve solutions quickly, systematically mutating the worst individuals. At the same time preserving throughout the search process the possibility of probing different regions of the design space (via avalanches), enabling the algorithm to escape local optima [1]. The basic EO algorithm showed good performance on problems like graph partitioning but when applied to other types of problems, it led to a deterministic search [1]. To overcome this, the algorithm was modified and an adjustable parameter introduced, so that the algorithm could escape local optima. This implementation of the EO algorithm received the name $\tau$-EO algorithm and showed superior performance to the standard implementation, even in cases where the basic EO algorithm would not lead to "dead ends".

The GEO algorithm uses the same approach of the $\tau$-EO, but the way it is implemented allows it to be applied readily in a broad class of engineering problems. The algorithm is of easy implementation, does not make use of derivatives and can be applied to either unconstrained or constrained problems. Moreover, it can deal in principle with any kind of variable, either continuous, discrete or integer. This make it suitable to be used in complex inverse design problems, where traditional gradient methods could not be applied properly due to, for example, non-convexities or use of mixed types of design variables. In the GEO, the design variables are encoded as binary strings and a fitness value is assigned to each bit. The algorithm mutates the bits trying to find the configuration that gives the best value for the objective function. It is implemented as follows:

1. Initialize randomly a binary string of length L that encodes N design variables of bit length L/N.
2. For the current configuration C of bits, calculate the objective function value V and set $C_{best}$ = C and $V_{best}$ = V.
3. For each bit i do,
   a) flip the bit (from 0 to 1 or 1 to 0) and calculate the objective function value $V_i$ of the string configuration $C_i$,
   b) set the bit fitness $F_i$ as ($V_i$ - R), where R is a constant. It serves only as a reference number and can assume any value. The bit fitness indicates the relative gain (or loss) that one has in mutating the bit.
   c) return the bit to its original value.
4. Rank the N bits according to their fitness values, from k = 1 for the least adapted bit to k = N for the best adapted. In a minimization problem higher values of $F_i$ will have higher ranking. Otherwise for maximization problems. If two or more bits have the same fitness, rank them in random order, but following the general ranking rule.
5. Chose a bit i to mutate according to the probability distribution $P_{i\_k} = k^{-\tau}$, where $\tau$ is an adjustable parameter.
6. Set C = $C_i$ and V = $V_i$.
7. If $F_i < F_{best}$ ($F_i > F_{best}$, for a maximization problem) then set $F_{best} = F_i$ and $C_{best} = C_i$.
8. Repeat steps 3 to 7 until a given stopping criteria is reached.
9. Return $C_{best}$ and $F_{best}$.

Constraints to the objective function can be easily incorporated to the algorithm simply setting a high fitness value to the bit that, when flipped, leads the configuration to an unfeasible region of the design space. Note that the move to an infeasible region is not prohibited, since any bit has a chance to mutate according to the $P_{i,k}$ distribution. Moreover, no special condition is posed for the beginning of the search process, it can even start from an infeasible region.

A slightly different implementation of the GEO algorithm can be done changing the way the bits are ranked. Instead of ranking all the bits according to steps 3-4, we can rank them separately for each variable. In this way the bits of each variable "j" will have a rank "$k_j$" ranging from 1 to L/N. Now the mutation is done simultaneously for all design variables. That is, in step 5 one bit of each variable "j" is flipped according to the probability distribution $P_{j,i,k} = k_j^{-\tau}$. We will call this implementation hereinafter as $GEO_{var}$.

**Results**

The GEO and the $GEO_{var}$ algorithms were applied to a set of test functions described in [3]. They are multimodal multidimensional unconstrained functions with variables bounded by side constraints. As in the GAs used in [3], each variable is encoded in a binary string of 16 bits. All functions have one global optimum, where the value of the objective function is zero. In Figures 1 to 5 below, the performance of the GEO algorithms for the set of test functions is shown together with the results for the GAs. All data points on the graphs below represent an average of 50 independent runs. The best objective function value found in the search is shown against the number of function evaluations.

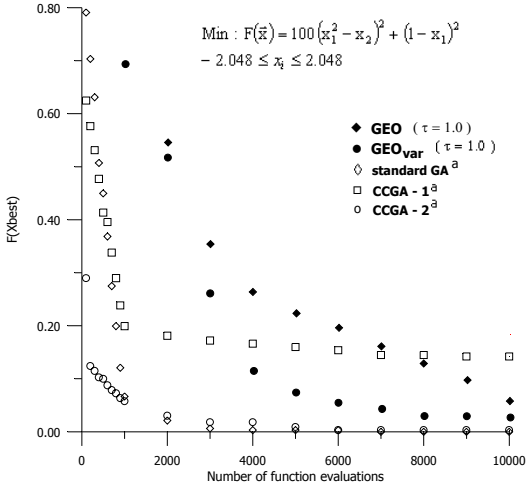


Figure 1 – Results for the Rosenbrock function. [a]From [3].

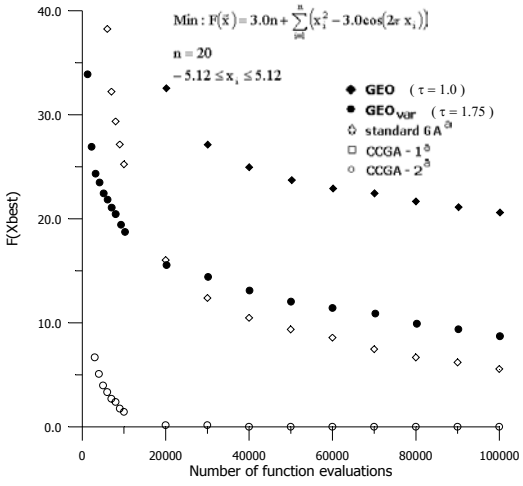


Figure 2 – Results for the Rastringin function. [a]From [3].

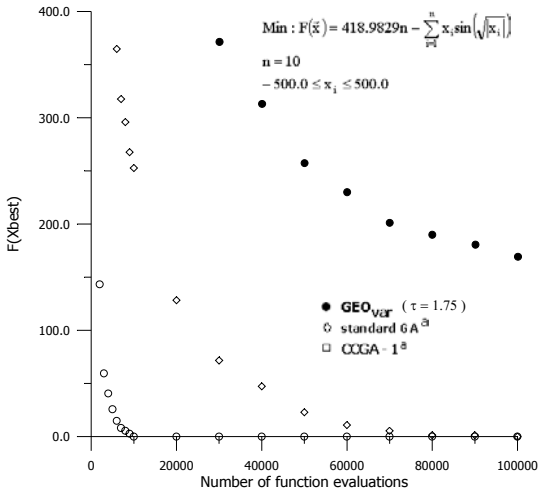


Figure 3 – Results for the Schwefel function. [a]From [3].

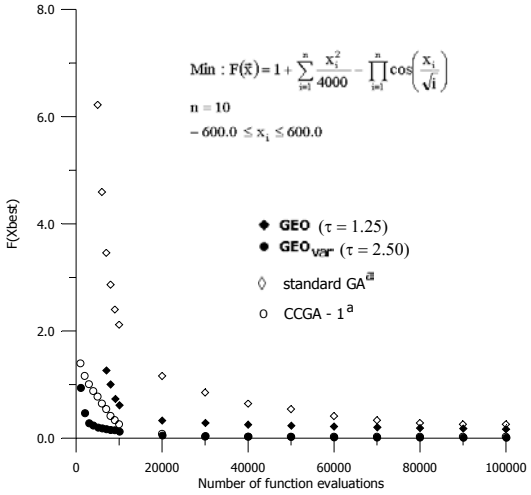


Figure 4 – Results for the Griewangk function. [a]From [3].

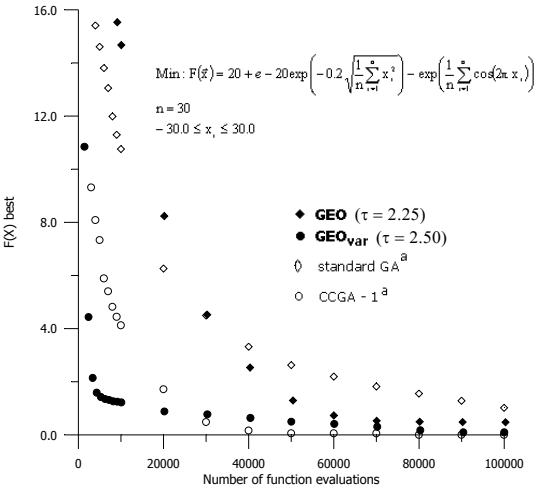


Figure 5 – Results for the Ackley function. [a]From [3].

## Conclusions

The results shown above indicate that the GEO can work successfully. Although it performed very poorly for the Rastrigin function, when compared to the GAs, it was quite competitive for the other test functions, mainly when the variables were tackled simultaneously ($GEO_{var}$). In fact, it must be remembered that does not exists a "best of all" optimization algorithm [5], and it is not expected that the GEO algorithm would outperform all the other kinds of stochastic algorithms in all cases. Rather, the present study intends to show that it is a potential candidate to be incorporated into the designer's tool suitcase. Ongoing research is aimed at the study of the implementation of the GEO algorithm to constrained function optimization and application to real inverse design problems.

## References

[1] Boettcher, S. and Percus, A.G. Optimization with Extremal Dynamics. *Physical Review Letters*, Vol. 86, pp. 5211-5214, 2001.

[2] Bak, P. and Sneppen, K. Punctuated Equilibrium and Criticality in a Simple Model of Evolution. *Physical Review Letters*, Vol. 71, Number 24, pp. 4083-4086, 1993.

[3] Potter, A.P. and De Jong, K.A. A Cooperative Coevolutionary Approach to Function Optimization. *The Third Problem Solving From Nature*, Springer-Verlag, pp. 249-257, 1994.

[4] Bak, P. *How Nature Works*, Copernicus, Springer-Verlag, 1999.

[5] Wolpert, D.H. and Macready, W.G. No Free Lunch Theorems for Search, *Santa Fe Institute Technical Report*, SFI-TR-95-02-010, 1995.