4th International Conference on Inverse Problems in Engineering
Rio de Janeiro, Brazil, 2002

Function Optimization Using Extremal Dynamics

Fabiano Luis de Sousa
INPE-DMC
Av. dos Astronautas, 1758
S.J.Campos, 12227-010, Brazil
Email: fabiano@dem.inpe.br

Fernando Manuel Ramos
INPE-LAC
Av. dos Astronautas, 1758
S.J.Campos, 12227-010, Brazil
Email: fernando@lac.inpe.br

ABSTRACT

In this paper a new stochastic algorithm for function optimization is presented. Called Generalized Extremal Optimization, it was inspired by the theory of Self-Organized Criticality and is intended to be used in complex inverse design problems, where traditional gradient based optimization methods may become inefficient. Preliminary results from a set of test functions show that this algorithm can be competitive to other stochastic methods such as the genetic algorithms.

NOMENCLATURE

k Index of bit rank.
L Length of binary string that encodes the design variables.
l Length of binary string for one design variable.
N Number of design variables.
V Value of the objective function for a given binary string.
x Design variable.
ΔV Bit fitness.
τ Free adjustable parameter of the optimization algorithm.

INTRODUCTION

Stochastic algorithms inspired by nature have been successfully used for tackling optimization problems in engineering and science. Simulated Annealing (SA)[1] and Genetic Algorithms (GAs)[2] are probably the two methods most used. Their robustness and ability to be easily implemented to a broad class of problems, regardless of such difficulties as the presence of multiple local minima in the design space and the mixing of continuous and discrete variables, has made them good tools to tackle complex problems, for example, in the aerospace field[3-7]. The main disadvantage of these methods is that they usually need a great number of objective function evaluations to be effective. Hence, in problems where the calculation of the objective function is very time consuming, these methods may become impracticable. Nevertheless, the availability of fast computing resources or the use of hybrid techniques[8-10] has made the power of those algorithms available even to that kind of problems. There are today many derivatives of the SA and GAs methods, created to give more efficiency to the proposed original algorithms, but that keep essentially their same principles.

Recently, Boettcher and Percus[11] have proposed a new optimization method based on a simplified model of biological evolution developed to show the emergence of Self-Organized Criticality (SOC) in ecosystems.[12] Called Extremal Optimization (EO), it has been successfully applied to tackle hard problems in combinatorial optimization.

Although algorithms such as SA, 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.

In this paper the Generalized Extremal Optimization (GEO) algorithm is presented. The GEO algorithm is built over the EO method, but the way it is implemented allows it to be readily applied to a broad class of engineering problems. The algorithm is of easy implementation, does not make use of derivatives and can be applied to nonconvex or disjoint problems. It can also deal in principle with any kind of variable, either continuous, discrete or integer. All these features make it suitable to be used in complex inverse design problems, where traditional gradient methods could not be applied properly due to, for example, the presence of multiple local minima or use of mixed types of design variables. In this work the performance of the GEO algorithm is tested in a set of non-linear multimodal functions used commonly to test GAs. The performance of the GEO algorithm for these functions is compared with the ones for a standard GA and the Cooperative Co-

evolutionary GA (CCGA) proposed by Potter and De Jong.[13]

THE EXTREMAL OPTIMIZATION ALGORITHM

Self-organized criticality has been used to explain the behavior of complex systems in such different areas as geology, economy and biology.[14] The theory of SOC states that large interactive 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. The probability distribution of the sizes "s" of these avalanches is described by a power law in the form $P(s) \sim s^{-\gamma}$, where γ is a positive parameter. That is, smaller avalanches are more likely to occur than big ones, but even avalanches as big as the whole system may occur with a non-negligible probability. To show that SOC could explain features of systems like the natural evolution, Bak and Sneepen[12] developed a simplified model of an ecosystem in which species are placed side by side on a line with periodic boundary conditions. To each species, a fitness number is assigned randomly, with uniform distribution, in the range [0,1]. The least adapted species, the one with the least fitness, is then forced to mutate, and a new random number assigned to it. The change in the fitness of the least adapted species alters the fitness landscape of their neighbors, and to cope with that new random numbers are also assigned to them, even if they are well adapted. After some iterations, the system evolves to a critical state where all species have fitness above a critical threshold. However, the dynamics of the system eventually causes a number of species to fall below the critical threshold in avalanches that can be as big as the whole system.

An optimization heuristic based on a dynamic search that embodies SOC would evolve solutions quickly, systematically mutating the worst individuals. At the same time this approach would preserve throughout the search process, the possibility of probing different regions of the design space (via avalanches), enabling the algorithm to escape local optima. Inspired by the SOC theory, the basic EO algorithm was proposed as follows:[11]

1.Initialize configuration C of design variables x_i at will; set $C_{best} = C$.

2. For the current configuration C,
   a) set a fitness F_i to each variable x_i,
   b) find j satisfying $F_j \leq F_i$ for all i,
   c) choose C' in a neighborhood N(C) of C so that x_j must change,
   d) accept C = C' unconditionally,
   e) if $F(C) < F(C_{best})$ then set $C_{best} = C$.

3. Repeat step (2) as long as desired.

4. Return C_{best} and $F(C_{best})$.

The above algorithm shows good performance on problems, such as graph partitioning, where it can choose new configurations randomly among neighborhoods of C, while satisfying step 2c. But when applied to other types of problems, it can lead to a deterministic search.[11] To overcome this, the algorithm was modified as follows: in step 2b the N variables x_i are ranked so that to the variable with the least fitness is assigned rank 1, and to the one with the best fitness rank N. Each time the algorithm passes through step 2c a variable is chosen to be mutated according to a probability distribution of the k ranks, given by:

$$P(k) = k^{-\tau}, \quad 1 \leq k \leq N, \tag{1}$$

where τ is a positive adjustable parameter. For $\tau \to 0$, the algorithm becomes a random walk, while for $\tau \to \infty$, we have a deterministic search. The introduction of the parameter τ, allows the algorithm to choose any variable to mutate, but privileging the ones with low fitness. This implementation of the EO method received the name τ-EO algorithm[11], and showed superior performance to the standard implementation even in cases where the basic EO algorithm would not lead to local minima.

As pointed out by Boettcher and Percus,[11] "a drawback of the EO method is that a general definition of fitness for the individual variables may prove ambiguous or even impossible". What means that for each new optimization problem assessed, a new way to assign the fitness to the design variables may have to be created. Moreover, to our knowledge it has been applied so far to combinatorial problems with no implementation to continuos functions. In order to make the EO method applicable to a broad class of design optimization problems, without concern to how the fitness of the design variables would be assigned and capable to tackle either continuos, discrete or integer variables, a generalization of the EO, called Generalized Extremal Optimization, was devised. In this new algorithm, the fitness assignment is not done directly to the design variables, but to a "population of species" that encodes the variables. Each species receives its fitness, and eventually mutates, following general rules. The GEO algorithm is described in the next Section.

THE GENERALIZED EXTREMAL OPTIMIZATION ALGORITHM

We devised the GEO algorithm using the same logic of the evolutionary model of Bak and Sneppen,[12] but applying the τ-EO approach to choose the species that will mutate. Following Bak and Sneppen,[12] L species are aligned and for each species is assigned a fitness value that will determine the species that are more prone to mutate. We can think of these species as bits that can assume the values of 0 or 1. Hence, the entire population would consist of a single binary string. The design variables of the optimization problem are encoded in this string that would be similar to a chromosome in a canonical GA, but with each bit considered as a species or individual, as shown in Figure 1.

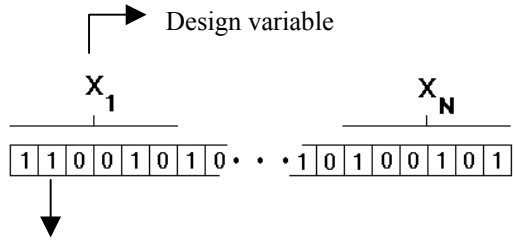

Figure 1 – Encoding of N design variables. In this example each design variable is represented by 6 bits.

To each species (bit) is assigned a fitness number that is proportional to the gain (or loss) the objective function value has in mutating (flipping) the bit. All bits are then ranked from rank 1, for the least adapted bit, to N for the best adapted. A bit is then chosen to mutate (flip) according to the probability distribution (1). This process is repeated until a given stopping criteria is reached and the best configuration of bits (the one that gives the best value for the objective function) found through the process is returned. In Figure 1

The practical implementation of the GEO algorithm to a function optimization problem is as follows:

1. Initialize randomly a binary string of length L that encodes N design variables of bit length l_j (j = 1, N). For the initial configuration C of bits, calculate the objective function value V and set C_{best} = C and V_{best} = V.

2. For each bit i of the string, at a given iteration:
   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 as $\Delta V_i = (V_i - V_{best})$. It indicates the relative gain (or loss) that one has in mutating the bit, compared to the best objective function value found so far.
   c) return the bit to its original value.

3. Rank the bits according to their fitness values, from k = 1 for the least adapted bit to k = L for the best adapted. In a minimization problem, higher values of ΔV_i will have higher ranking, and otherwise for maximization problems. If two or more bits have the same fitness, rank them randomly.

4. Choose with equal probability a candidate bit i to mutate. Generate a random number RAN with uniform distribution in the range [0,1]. If the mutating probability $P_i(k)$ of the chosen bit is equal or greater than RAN the bit is confirmed to mutate. Otherwise, the process is repeated until a bit is confirmed to mutate.

5. For the bit i chosen to mutate set $C = C_i$ and $V = V_i$.

6. If $V < V_{best}$ ($V > V_{best}$, for a maximization problem) then set $V_{best} = V$ and $C_{best} = C$.

7. Repeat steps 2 to 6 until a given stopping criteria is reached.

8. Return C_{best} and V_{best}.

Equality and inequality constraints can be easily incorporated to the algorithm simply setting a high (for a minimization problem) or low (for a maximization problem) fitness value to the bit that, when flipped, leads the configuration to an unfeasible region of the design space. Side constraints are directly applied through the encoding of the design variables. Note that the move to an infeasible region is not prohibited, since any bit has a chance to mutate according to the P(k) distribution. Moreover, no special condition is posed for the beginning of the search process, which can even start from an infeasible region.

A slightly different implementation of the GEO algorithm can be obtained, changing the way the bits are ranked and mutated. Instead of ranking all the bits according to steps 2-3, we can rank them separately for each variable. In this way the bits of each variable will have a rank ranging from 1 to l_j. In step 4 one bit of each variable is chosen to be flipped according to the probability distribution P(k). We will call this implementation hereinafter GEO_{var}. In the following Section the performance of the GEO algorithm is verified against a set of test functions.

RESULTS

The GEO algorithm and its variation GEO_{var} were applied to a set of test functions described in [13]. They are nonlinear, multimodal, multidimensional functions with variables bounded by side constraints. As in the GAs used in [13], each variable is encoded in 16 bits. All functions have one global optimum, where the value of the objective function is zero. As with any stochastic algorithm, the performance of GEO is influenced by its control parameter. In order to find the "best" value of τ applicable for each test function, we varied τ in the range [0.25,3.0] with steps of 0.25. For a given test function, the best value of τ was the one that lead to the best (minimal) value for the objective function, after a given number of function evaluations (NFE).

In Figures 2 to 6, 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 through the search is shown against the number of function evaluations.

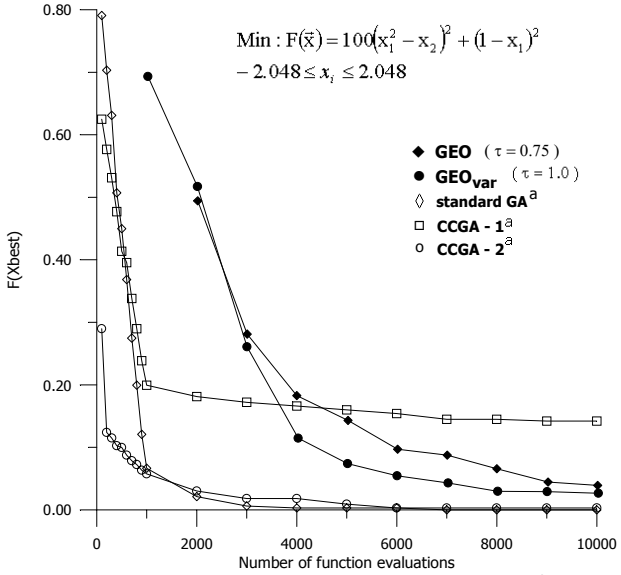

Figure 2 – Results for the Rosenbrock function. [a]From [13].

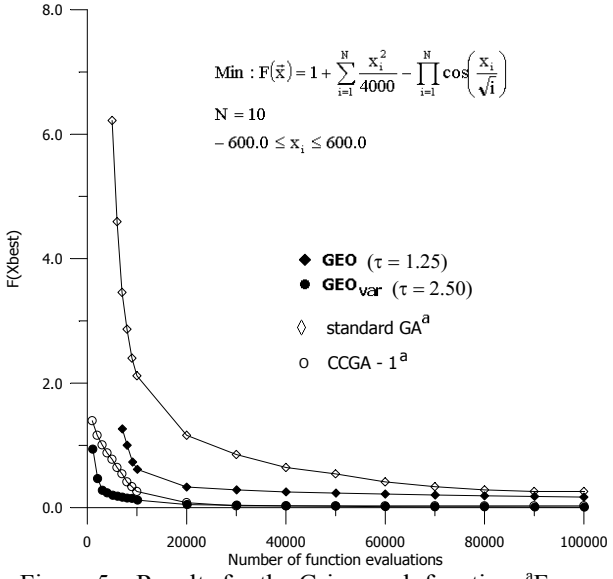

Figure 5 – Results for the Griewangk function. [a]From [13].

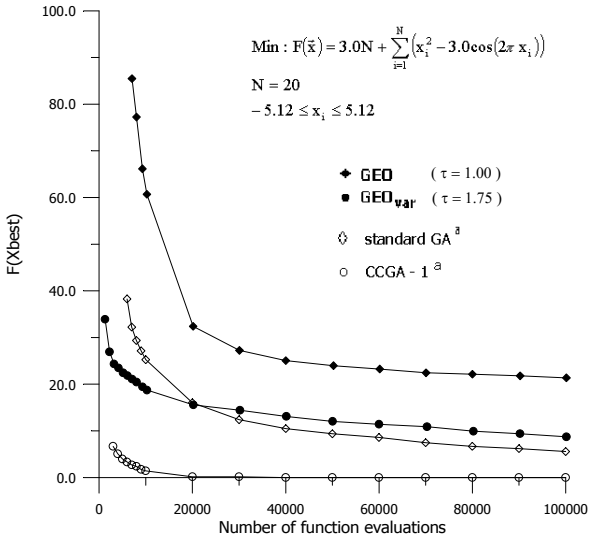

Figure 3 – Results for the Rastringin function. [a]From [13].

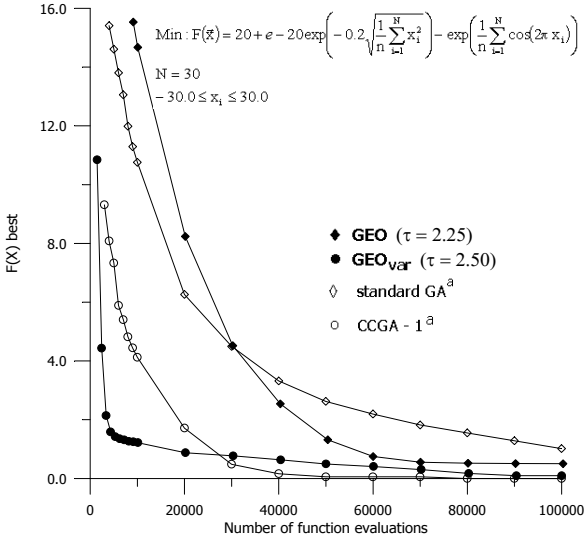

Figure 6 – Results for the Ackley function. [a]From [13].

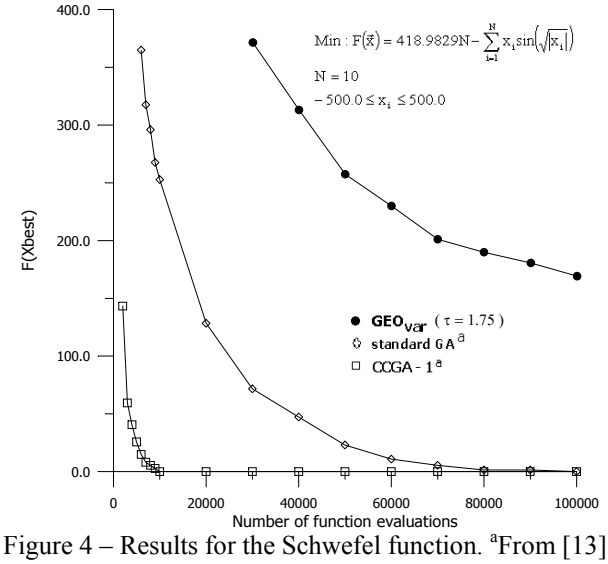

Figure 4 – Results for the Schwefel function. [a]From [13].

From the results shown throughout this Section, it can be seen that the GEO_{var} performed equally or better than the GEO for all functions. This indicates that, at each iteration, mutating one bit per variable may be advantageous compared to mutating only one bit for the whole string.

It can be also observed that, for a given test function, the value of τ that gave the best results was always lesser in the GEO algorithm than in the GEO_{var}. It must be also remarked, that the range where the "best" τ was found for both GEOs is not large, what means that the computational effort to "fine tune" τ is not really a burden for the method.

Finally, the results shown above indicate that the GEO can work successfully. Although it performed very poorly for the Schwefel 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,[15] and it is not expected that the GEO algorithm would outperform all the other kinds of stochastic algorithms in all cases.

CONCLUSIONS

In this paper the Generalized Extremal Optimization algorithm was presented. Inspired by the theory of Self-Organized Criticality, it is an stochastic algorithm devised to tackle complex design optimization problems that presents such features as nonconvex design spaces or presence of different kinds of design variables. As an "a priori" advantage over other popular stochastic algorithms, it has only one adjustable parameter, and can be easily fine tuned to give its best performance on a given problem. Tested in a set of nonlinear, multimodal functions commonly used to assess the performance of stochastic algorithms, it showed to be 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 its application to real inverse design problems.

ACKNOWLEDGEMENTS

F. M. Ramos acknowledges the support of CNPq-Brazil through the research grant 300171/97-8.

REFERENCES

1. Kirkpatrick, S., Gellat, C.D. and Vecchi, M.P., Optimization by Simulated Annealing, *Science*, Vol. 220, Number 4598, 1983, pp. 671-680.

2. Goldberg, D.E., *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Publishing Company, 1989.

3. Ahmed, Q., Krishnakumar, K. and Neidhoefer, J., Applications of evolutionary Algorithms to Aerospace Problems – A Survey, *Computational Methods in Applied Sciences'96*, John Wiley & Sons, 1996, pp. 237-242.

4. Schoonover, P.L., Crossley, W.A. and Heister, S.D., Application of a Genetic Algorithm to the Optimization of Hybrid Rockets, *Journal of Spacecraft and Rockets*, Vol. 37, No. 5, 2000, pp. 622-629.

5. Jones, B.R., Crossley, W.A. and Lyrintzis, A., Aerodynamic and Aeroacoustic Optimization of Rotocraft Airfoils via a Parallel Genetic Algorithm, *Journal of Aircraft*, Vol. 37, No. 6, 2000, pp. 1088-1096.

6. Wang, X. and Damodaran, M., Aerodynamic Shape Optimization Using Computational Fluid Dynamics and Parallel Simulated Annealing Algorithms, *AIAA Journal*, Vol. 39, No. 8, 2001, pp. 1500-1508.

7. Jilla, C.D. and Miller, D.W., Assessing the Performance of a Heuristic Simulated Annealing Algorithm for the Design of Distributed Satellite Systems, *Acta Astronautica*, Vol. 48, No. 5-12, 2001, pp. 529-543.

8. Vicini, A. and Quagliarella, D., Airfoil and Wing Design Through Hybrid Optimization Strategies, *AIAA Journal*, Vol. 37, No. 5, pp. 634-641, 1999.

9. Crain, T. Bishop, R.H. and Fowler, W., Interplanetary Flyby Mission Optimization Using a Hybrid Global-Local Search Method, *Journal of Spacecraft and Rockets*, Vol. 37, No. 4, pp. 468-474, 2000.

10. Desai, R. and Patil, R., SALO: Combining Simulated Annealing and Local Optimization for Efficient Global Optimization, *Los Alamos National Laboratory, TR LA-UR-95-2862*, Albuquerque, NM, 1995.

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

12. 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.

13. 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.

14. Bak, P. *How Nature Works*, Copernicus, Springer-Verlag, 1999.

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