An Evolutionary Algorithm for Minimizing Multimodal Functions

D.G. Sotiropoulos, V.P. Plagianakos and M.N. Vrahatis
University of Patras, Department of Mathematics,
Division of Computational Mathematics & Informatics,
GR-265 00, Patras, Greece.
e-mail: dgs|vpp|vrahatis@math.upatras.gr.

Abstract

A new evolutionary algorithm for the global optimization of multimodal functions is presented. The algorithm is essentially a parallel direct search method which maintains a populations of individuals and utilizes an evolution operator to evolve them. This operator has two functions. Firstly, to exploit the search space as much as possible, and secondly to form an improved population for the next generation. This operator makes the algorithm to rapidly converge to the global optimum of various difficult multimodal test functions.

1 Introduction

In this contribution we propose a new evolutionary algorithm for global minimization of non-linear and non-differentiable continuous functions. More formally, we consider the following general global optimization problem:

$$f^* = \min_{x \in \mathcal{D}} f(x),$$

where $f : \mathcal{D} \to \mathbb{R}$ is a continuous function and the compact set $\mathcal{D} \subset \mathbb{R}^n$ is an n-dimensional parallelepiped.

Often, analytic derivatives of f are unavailable particularly in experimental settings where evaluation of the function means measuring some physical or chemical quantity or performing an experiment. Inevitably, noise contaminates the function values and as a consequence, finite difference approximation of the gradient is difficult, if not impossible, the procedure to estimate the gradient fails, and methods that require derivatives are precluded. In such cases direct search approaches are the methods of choice.

If the objective function is unimodal in the search domain, one can choose among many good alternatives. Some of the available minimization methods involve only function values, such as those of Hook and Jeeves [2] and Nelder and Mead [5]. However, if the objective function is multimodal, the number of available algorithms is reduced to very few. In this case, the algorithms quoted above tend to stop at the first minimum encountered, and cannot be used easily for finding the global one.

Search algorithms which enhance the exploration of the feasible region through the use of a population, such as Genetic Algorithms (see, [4]), and probabilistic search techniques such as Simulated Annealing (see, [1, 7]), has been the object of many publications. In analogy with the behavior of natural organisms, random search algorithms have often called *evolutionary methods*. The generation of a new trial point corresponds to mutation while a step toward the minimum can be viewed as selection.

In the present work we propose an evolutionary algorithm in which we enhance the exploration of the feasible region utilizing a specially designed evolution operator, called Simplex Operator, and based on the idea of simplex search method. It starts with a specific number of N dimensional candidate solutions, as an initial population, and evolves them over time, in such a way that each new iteration of the algorithm gives a better population.

The remaining of the paper is organized as follows: In Section 2 we present our Simplex Evolution algorithm (SE) for minimizing multimodal functions. The algorithm has been tested against the Annelead Nelder and Mead simplex method and the corresponding results are presented in Section 3. The final section contains concluding remarks and a short discussion for further work. In the appendix we list the problems used in our tests from Levy et al. [3].

2 The Simplex Evolution

Simplex Evolution is based on the collective learning process within a population of individuals, each of which represents a search point in the space of potential solutions to a given problem. The population is chosen randomly and should try to cover the entire search space $\mathcal{D}$, and evolves towards better and better regions of the search space by means of deterministic steps that use randomly selected individuals. The number of the candidate solutions does not change during the minimization process. At each iteration, called *generation*, the evolution of the population occurs through the use of an operator called *mutation*.

The key idea is to construct a mechanism, which at each generation takes each individual of the population, and replace it with a better one for the next generation. To this end, we have specially redesign the traditional mutation operator in order to enhance the process of the exploitation of the search space, in such a manner that typically avoids getting stuck in local minimum points. Our mutation operator, called *Simplex Operator*, is based on a simplex search, which is a geometrical figure consisting in N dimensions of $N+1$ points and all interconnecting lines and faces. In general we are interested only in simplexes that are nondegenerate, i.e., that enclose a finite inner N–dimensional volume. For each candidate solution, called *BasePoint*, of the current generation, we choose N other candidates and we form a simplex. If we assume that the *BasePoint* of a nondegenerate simplex is taken as the origin, then the N other points define vector directions that span the N–dimensional vector space.

In the sequence, for this simplex we perform two of the three possible trial steps: a reflection, an expansion or a contraction step, in order to approach the minimum by moving away from high values of the objective function, without moving out of the search domain. The current point is replaced in the next generation by the resulted mutated one. This last operation is called *selection* and produces an improved population for the next generation.

2.1 The Simplex Operator

The Simplex Operator is a specially designed evolution operator with deterministic steps that enhances the exploitation of the search space. This operator amounts to replacing a single individual, as a description of the system state, by a simplex of $N+1$ points. In practice, we are only interested in nondegenerate simplexes. The *moves* of the simplex are the same as described in [5, 6], namely reflection, expansion, and contraction. The least preferred point of the simplex is reflected through or contracted towards the center of the opposite face of the simplex. Actually, this operator performs only one *cycle* of the Simplex method as shown in the following algorithm which gives the basic steps of the operator.

In line 3 we check if the simplex $\mathcal{S}$ is degenerate, regarding to the function values of its vertices. Specifically, we check if the standard deviation of the function at the $N+1$ vertices of the

```
    SIMPLEXOPERATOR(BasePoint x_1)
 1: Select points x_i, where i = 2, ..., N + 1;
 2: Construct the simplex S = ⟨x_1, x_2, ..., x_{N+1}⟩;
 3: if S is degenerate go to 1;
 4: Rank x_i based on the function values;
 5: Reflect(S) → x_r;
 6: if f(x_r) < f(x_1) then
 7:    Expand(S) → x_e;
 8:    if f(x_e) < f(x_r) then
 9:       return the expansion point x_e;
10:    else
11:       return the reflection point x_r;
12:    end if
13: else
14:    Contract(S) → x_c;
15:    if f(x_c) < f(x_r) then
16:       return the contraction point x_c;
17:    else
18:       return the best vertex of the simplex S;
19:    end if
20: end if
```

current simplex is smaller than some prescribed small quantity ε, i.e.

$$\sigma = \left\{ \sum_{i=1}^{N+1} \frac{[f(x_i) - f(x_0)]^2}{N+1} \right\}^{1/2} \leq \varepsilon, \tag{1}$$

where x_0 is the centroid of the simplex $\mathcal{S}$. In the implementation of the algorithm we have used $\varepsilon = 10^{-15}$. There is no case of an infinite loop between the lines 1 and 3, where the simplex $\mathcal{S}$ is examined for degeneracy, since in the SE algorithm (see Section 2.2), the (1) has been enforced as a termination criterion for the entire population. Therefore, we can form at least one non-degenerate simplex. Using (1) we avoid unnecessary function evaluations.

To guarantee convergence in the feasible region $\mathcal{D}$, we have constrained the operations reflection and/or expansion (lines 5 and 7), in a such a way that the returned point is always in $\mathcal{D}$. If the returned point is outside of the feasible region $\mathcal{D}$, we subdivide the prefixed values of reflection and/or expansion constants until the new point is included in $\mathcal{D}$.

If the simplex operator fails to improve the base point x_1 during the reflection, expansion or contraction operations then we return the best vertex of the simplex $\mathcal{S}$. Simplex Operator has embedded (in lines 9, 11, 16, 18) the selection step, as it returns always a better individual in order to replace the *BasePoint*. This is consistent with the idea of selection in evolution algorithms, since the "good" individuals are reproduced more often than the worse ones in the next generation. In contrast with other evolution algorithm, SE deterministically choose the individuals for the next generation.

2.2 The SE Algorithm

In this subsection we present our evolutionary algorithm (SE), which maintains a population of *PopSize* individuals and utilizes the Simplex Operator to evolve them over time. The algorithm

does not have any control parameters, as other global optimization methods. The user must supply only the search region (a box) and the population size *Popsize*. In our implementation we have chosen the *Popsize* to be five times the dimension of the problem to be optimized. As the population iterates through successive generations, the individuals tend toward the global optimum of the objective function f. To generate a new population $G(t+1)$ on the basis of the previous one $G(t)$ the algorithm SE performs the steps as shown in the pseudo code.

	SE ALGORITHM
1:	$t = 0$;
2:	Generate the initial population $G(t)$;
3:	Evaluate the initial population $G(t)$;
4:	**while** termination criterion not reached **do**
5:	**for** $i = 1, \ldots, PopSize$ **do**
6:	$BasePoint \leftarrow x_i \in G(t)$;
7:	$x_i^{new} \leftarrow$ SimplexOperator($BasePoint$);
8:	Put x_i^{new} in the next generation $G(t+1)$;
9:	**end for**
10:	$t = t + 1$;
11:	**end while**

Classical termination criteria for the evolution algorithms are: a) maximum number of generations, and b) maximum number of function evaluations. Obviously, if the global minimum value is known, then an efficient termination criterion can be defined; the method is stopped when finds the minimizer within a predetermined accuracy. This happens, for example, in the case of the non–linear least squares.

Except the previous ones we can enforce an additional termination criterion, since SE is based on the simplex method. This criterion checks at each generation if the entire population has converged, examining the Relation (1). If the standard deviation of the function values is smaller than ε, then this is a strong indication that further search is impossible, since we cannot form simplexes big enough to exploit the search space. Therefore, we can stop the algorithm, or possibly, rank the population and restart the algorithm by reinitializing the worst individuals. By doing this we enrich the population and give the algorithm a chance for further exploitation of the search space.

3 Tests and Results

In this section we present some preliminary results of the SE method. We decided to test the effectiveness of the SE method against the Annealed Nelder and Mead method (ANM) [6], since it has few, easy tunable, control parameters, in contrast with other well-known pure annealing methods that have many control variables [1, 7]. ANM has been chosen also owing to its ability to accept downhill as well as uphill steps. The probability to accept uphill steps decreases with the "temperature" of the system, which in turn decrease with time. This mechanism enables ANM to escape local minima when the temperature does not tend to zero. When the annealing part is switched off, the simplex method remains. We have not chosen a gradient method, since the test functions are multimodal and such a method would immediately get trapped at the first local minimum encountered.

To evaluate the performance of the algorithms we have tested them in seven well-known multimodal test functions from Levy et. al. (see Appendix). The admissible domains of the

test functions were $[-10, 10]^n$, where n is the dimension of the problem. The test for ANM has been made starting from 100 regular simplexes randomly generated in the region $\mathcal{D}$. The SE method, since it maintains a population, has been initialized 100 times, each time with a different randomly chosen population. We retained two comparison criteria: the success rate to find the global minimum, and the average number of function evaluations, during 100 independent runs. We consider that the method has succeeded, if it has found the global minimum with accuracy 10^{-3}. It must be noted that when a method fails to find the global minimum its function evaluations are not counted.

Table 1: Minimization of seven test functions.

		SE		ANM	
	N	FE	Succ.	FE	Succ.
Levy No. 3	2	934	89%	135	32%
Levy No. 5	2	547	86%	116	9%
Levy No. 8	3	325	100%	138	57%
Levy No. 9	4	546	100%	160	45%
Levy No. 10	5	450	100%	186	16%
Levy No. 11	8	4404	100%	–	0%
Levy No. 12	10	11619	100%	–	0%

In Table 1 one can observe that SE is more costly compared to the ANM, but excibited better success rate and the number of function evaluations needed is quite predictable regarding to the number of the local minima. On the other hand, the ANM was unable to find the global minimum in high-dimensional problems, such as Levy No. 11 and No. 12, where a multitude of local minima (10^8 and 10^{10} respectively) exist. Summing up, the results show that the SE method is a predictable and reliable method, in contrast with the annealed version of simplex method, especially in high dimensions.

4 Conclusion

Preliminary results indicate that this new evolutionary algorithm fulfills our promises, since it exhibits high performance and good convergence properties in consecutive independent trials. It can be applied to nondifferentiable, nonlinear and high-dimensional multimodal objective functions. Moreover, the method is heuristic-free, i.e. does not uses any control parameters.

In a recent work [8], we have successfully hybridize a genetic algorithm with an interval branch-and-bound one, for global optimization. In its present form, SE gives us one global minimum, without guarantee. Our aim is to combine the SE with an interval global optimization algorithm in order to find *all* the global minimizers with certainty.

Future work will also include testing on bigger and more complex real-life optimization problems and comparison with other well-known global optimization methods, such as those found in [1, 7]. Last but not least, we will design and implement a parallel version of the algorithm, since SE is a parallel direct search, and it can be efficiently executed on a parallel machine or a network of computers.

References

[1] A. Corana, M. Marchesi, C. Martini, and S. Ridella, Minimizing Multimodal Functions of Continuous Variables with the "Simulated Annealing" Algorithm,*ACM Trans. on Math. Software*, **13**, 262–280, 1987.

[2] R. Hook, and T. Jeeves, Direct search solution of numerical and statistical problems, *J. ACM*, **7**, 212–229, 1969.

[3] A. Levy, A. Montalvo, S. Gomez, and A. Galderon, Topics in Global Optimization, Lecture Notes in Mathematics No. 909, Springer-Verlag, New York, 1981.

[4] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, Springer-Verlag, 1996.

[5] Nelder J.A. and Mead R., A simplex method for function minimization. *Computer J.*, **7**, 308–313, 1965.

[6] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, Numerical Recipes in FORTRAN, Second Edition, Cambridge University Press, 1992.

[7] P. Siarry, G. Berthiau, F. Durbin, and J. Haussy, Enhanced Simulated Annealing for Globally Minimizing Functions of Many Continious Variables, *ACM Trans. on Math. Software*, **23**, 209–228, 1997.

[8] D. Sotiropoulos, E. Stavropoulos, and M. Vrahatis, A New Hybrid Genetic Algorithm for Global Optimization, Nonlinear Analysis, Theory, Methods & Applications, **30**, 4529–4538, 1997.

Appendix: List of test functions

(1) Levy No. 3, $(n = 2)$.

$$f(x) = \sum_{i=1}^{5} i \cos[(i+1)x_1 + i] \sum_{j=1}^{5} j \cos[(j+1)x_2 + j], \ -10 \leq x_i \leq 10$$

where $-10 \leq x_i \leq 10, i = 1, 2$. There are about 760 local minima and 18 global minima. The global minimum is $f^* = -176.542$.

(2) Levy No. 5, $(n = 2)$.

$$f(x) = \sum_{i=1}^{5} i \cos[(i+1)x_1 + i] \sum_{j=1}^{5} j \cos[(j+1)x_2 + j] + (x_1 + 1.42513)^2 + (x_2 + 0.80032)^2,$$

where $-10 \leq x_i \leq 10, i = 1, 2$. There are about 760 local minima and one global minimum $f^* = -176.1375$ at $x^* = (-1.3068, -1.4248)$. Levy No. 5 is identical to Levy No. 3 except for the addition of a quadratic term. The large number of local optimizers makes it extremely difficult for any approximation method to find the global minimizer.

(3) Levy No. 8, $(n = 3)$.

$$f(x) = \sin^2(\pi y_1) + \sum_{i=1}^{n-1} (y_1 - 1)^2 \ [1 + 10 \sin^2(\pi y_{i+1})] \ + (y_n - 1)^2$$

where $y_i = 1 + (x_i - 1)/4$, $(i = 1, \ldots, n)$ and $x_i \in [-10, 10]$, $i = 1, 2, 3$. There are about 125 local minima and one global minimum $f^* = 0$ at $x^* = (1, 1, 1)$. Levy No. 8 is difficult due to the combinations of different periods of the sine function.

(4) Levy No. 9, $(n = 4)$.
Same as problem (3) with $n = 4$ and $x_i \in [-10, 10]$, $i = 1, \ldots, 4$. There are about 625 local minima and one global minimum $f^* = 0$ at $x^* = (1, 1, 1, 1)$.

(5) Levy No. 10, $(n = 5)$.
Same as problem (3) with $n = 5$ and $x_i \in [-10, 10]$, $i = 1, \ldots, 5$. There are about 10^5 local minima and one global minimum $f^* = 0$ at $x^* = (1, \ldots, 1)$.

(6) Levy No. 11, $(n = 8)$. Same as problem (3) with $n = 8$ and $x_i \in [-10, 10]$, $i = 1, \ldots, 8$. There are about 10^8 local minima and one global minimum $f^* = 0$ at $x^* = (1, \ldots, 1)$.

(7) Levy No. 12, $(n = 10)$. Same as problem (3) with $n = 10$ and $x_i \in [-10, 10]$, $i = 1, \ldots, 10$. There are about 10^{10} local minima and one global minimum $f^* = 0$ at $x^* = (1, \ldots, 1)$.