


HYBRID DIFFERENTIAL EVOLUTION WITH THE TOPOGRAPHICAL HEURISTIC

Ane Élide Nogueira Frauches Almoia 

anefrauches@gmail.com

Wagner F. Sacco 

Instituto de Engenharia e Geociências, Universidade Federal do Oeste do Pará

wagner.sacco@ufopa.edu.br

Antônio J. Silva Neto 

Instituto Politécnico, Universidade do Estado do Rio de Janeiro

ajsneto@iprj.uerj.br

Abstract – In this article, we present a new hybrid differential evolution (DE) which employs a topographical heuristic introduced in the early nineties as part of a global optimization method. This heuristic is used to select individuals from the DE population in order to be starting points of instances of the Hooke–Jeeves algorithm. The solutions achieved in this phase are potential candidates for the next generation. The method, called TopoDE, is compared with other stochastic optimization algorithms using challenging benchmark problems. The results obtained are quite promising.

Keywords – Differential Evolution, Topographical Heuristic, Hybrid methods, Hooke–Jeeves method.

1 INTRODUCTION

Differential evolution [1] is a stochastic direct search populational algorithm, originally designed for continuous optimization, in which the optimality search is performed mutating, using crossover and selecting the individuals along the generations. DE is an evolutionary optimization algorithm that the mutation operator has a crucial role, being the main responsible for the algorithm's efficiency. In fact, this algorithm owes its name to mutation, as this mechanism is based on the difference vector between individuals of the current population. A great number of variations of the original DE have been proposed in the literature [2]. There have been many applications of differential evolution to practical problems [3–6].

In this article, we present a new variant of the differential evolution algorithm. This algorithm, called Hybrid Topographical Differential Evolution (TopoDE), is a hybridization of DE and the Hooke–Jeeves direct local search algorithm [7] which the topographical heuristic [8] is used to select the individuals of the former algorithm that will be initial solutions for instances of the latter. This heuristic is based on the topographical information on the objective function, and originally is part of an optimization method, the Topographical Algorithm [8–12].

Let's describe some algorithms from the literature that are similar to TopoDE. Ali and Törn [13] introduced the Topographical Differential Evolution (TDE), which employs the topographical heuristic within the DE framework in a different manner from our work. Their method uses two sets of populations. The first set is formed by the regular DE population. The second set, by its turn, is formed by points rejected by DE along the generations. These points form the topograph that will generate, after a determined number of generations, initial solutions for a local search algorithm. Then, a certain number of the final solutions obtained by local search replace the worst individuals from the first set. Caponio et al. [14] proposed the fast adaptive memetic algorithm (FAMA), which employs the Hooke–Jeeves and Nelder–Mead [15] local search methods executing the former only on the elite individual and the latter on 11 randomly selected individuals. Tirronen et al. [16] presented the Enhanced Memetic Differential Evolution (EMDE), which combines DE with three local search algorithms, among them Hooke–Jeeves. These local search algorithms are adaptively coordinated by means of a control parameter that measures fitness distribution among individuals of the population and a novel probabilistic scheme. Moser and Chiong [17] presented a new Hooke–Jeeves based Memetic Algorithm (HJMA) for dynamic function optimization, which is a hybridization of Hooke–Jeeves pattern search and Extremal Optimisation (EO), an optimization heuristic that was first introduced by Boettcher and Percus [18]. Recently, Domínguez–Isidro and Mezura–Montes [19] introduced an adaptive local search coordination for a multimeme Differential Evolution to constrained numerical optimization problems. The proposed approach associates a pool of direct local search operators within the standard Differential Evolution, among them the method of Hooke–Jeeves.

The remainder of the paper is organized as follows. In the next section, the canonical differential evolution is described. Section 3 provides a description of the topographical algorithm and of the method of Hooke–Jeeves followed by an exposition of the new method. In section 4, TopoDE is compared against other stochastic optimization algorithms using challenging benchmark problems. Finally, in section 5, conclusions are presented along with suggestions of further development.

2 THE CANONICAL DIFFERENTIAL EVOLUTION ALGORITHM

In this section, we describe the canonical version of differential evolution, as introduced by Storn and Price [1]. DE is applied to the minimization of an objective function $f(\mathbf{x})$, where \mathbf{x} is a continuous variable vector with domain $[\mathbf{low}, \mathbf{up}] \subset \mathbb{R}^n$.

Let's describe DE in a pseudo-code style. The algorithm is outlined in Algorithm 1 and its operators are described in Algorithms. 2, 3, 4, and 5.

Input parameters, which remain constant along the optimization process, are population size NP and, to be explained below, crossover rate CR and scaling factor F . First of all, an initial random population is generated by function “initialize”, as described in Algorithm 2. Note that each initial solution or individual must meet the boundary constraints. After that, inside a loop, the evolutive process starts until a stopping criterion is satisfied.

The first operation inside the loop is mutation, described by function “mutate”, Algorithm 3. In mutation, a trial solution is generated for each individual i as follows:

$$\hat{\mathbf{x}}_i = \mathbf{x}_{p(1)} + F(\mathbf{x}_{p(2)} - \mathbf{x}_{p(3)}), \quad (1)$$

where $p(1)$, $p(2)$, and $p(3)$ are random indexes mutually different from each other and different from index i , and F is a scaling factor in the range $[0, 2]$. The solution correspondent to the first random index, $\mathbf{x}_{p(1)}$, is known as the base vector. This vector is altered by the addition of the weighted difference of the two other solutions with indexes $p(2)$ and $p(3)$. The operation is repeated as long as trial solution $\hat{\mathbf{x}}_i$ is outside the domain.

After mutation, population goes through crossover, as in Algorithm 4. In this operation, component j of offspring \mathbf{y}_i is found from its parents \mathbf{x}_i and $\hat{\mathbf{x}}_i$ according to the rule

$$y_i^j = \begin{cases} \hat{x}_i^j, & \text{if } R^j \leq CR \text{ or } j = I_i, \\ x_i^j, & \text{otherwise,} \end{cases} \quad (2)$$

where I_i is a random integer in range $[0, n]$, R^j is a random in $[0, 1]$, and crossover rate CR , also in $[0, 1]$, controls the fraction of parameter values that are copied from the trial solution $\hat{\mathbf{x}}_i$. Note that alternative $j = I_i$ assures that at least one component will receive a mutated value.

Finally, there is the selection process, Algorithm 5, which defines the population of next generation as follows:

$$\mathbf{x}_i^{NIter+1} = \begin{cases} \mathbf{y}_i^{NIter}, & \text{if } f(\mathbf{y}_i^{NIter}) \leq f(\mathbf{x}_i^{NIter}). \\ \mathbf{x}_i^{NIter}, & \text{otherwise.} \end{cases} \quad (3)$$

The trial solution will only replace its counterpart in the current population if it's equal or better than the latter. As pointed out by Lampinen and Zelinka [20], in DE's selection scheme, a trial vector is not compared against all the individuals in the current population, but only against its counterpart.

Note that it's in function “select” (Algorithm 5), that the best solution found so far and its fitness value are stored.

As termination criterion, one may use the number of generations ($NIter$ in our pseudocode), the number of objective-function evaluations, or, as in Kaelo and Ali [21], $|f_{max} - f_{min}| < \varepsilon$, where f_{max} and f_{min} are the maximum and minimum function values within a generation.

Data: NP, CR, F .

Result: Optimal solution \mathbf{x}^* , $f(\mathbf{x}^*)$.

begin

$NIter \leftarrow 0$

$f(\mathbf{x}_i^*) \leftarrow 1.0E6$

initialize()

repeat

$NIter \leftarrow NIter + 1$

mutate()

crossover()

select()

until a termination criterion is satisfied

Algorithm 1: The Differential Evolution Algorithm.

3 HYBRID TOPOGRAPHICAL DIFFERENTIAL EVOLUTION

3.1 The topographical algorithm

Between the early seventies and mid-nineties, a global optimization paradigm based on clustering was studied by some researchers, mainly in Europe. The seminal article by Becker and Lago [22] was followed by, among others, Törn [23, 24],

Timmer [25], Törn and Viitanen [8, 9], and Ali and Storey [26]. Ali [27] and Levi and Haas [28] present fine reviews on the clustering methods. According to Törn and Zilinskas [29], the motivation for exploring clustering methods in based on the following:

- (a) It is possible to obtain a sample of points in the search space consisting of concentration of points in the neighborhood of local minimizers of the objective function f ;
- (b) The points in the sample can be clustered giving clusters identifying the neighborhoods of local minimizers and thus permitting local optimization methods to be applied.

The original TA algorithm is non-iterative and based on the exploration of the search space [26]. It consists of three steps [9]:

1. A uniform random sampling of N points in the search space;
2. The construction of the topograph, which is a graph with directed arcs connecting the accepted sampled points on a k -nearest neighbors basis, where the direction of the arc is towards a point with a larger function value. The minima of the graph are the points better than their neighbors, i.e., the nodes with no incoming arcs;
3. The topograph minima are starting points for a local optimization algorithm. The best point obtained from all the executions using each minimum as the initial solution is the result of the algorithm.

Originally, Törn and Viitanen [8, 9] obtained the initial solutions from step 1 sampling points in a unit hypercube, until N points with their nearest neighbors farther than a threshold distance δ were obtained. Then, these points were denormalized. But these authors mention that any other method that produces a very uniform covering can be used. In fact, they used the more efficient quasi-random sampling [30, 31] in an iterative version of TA [10]. In their tests, Törn and Viitanen [9] used mostly $N = 100$ or $N = 200$.

Step 2, the construction of the topograph, is the core of the method. First of all, an $N \times N$ symmetric distance matrix is computed. Following that, an $N \times k$ matrix called kNN -matrix is constructed containing, for each point, the indexes of its k -nearest neighbors sorted by distance. Next, this matrix, which is an undirected topograph, is transformed into a directed topograph indicating if the reference is to a point with larger or smaller objective function value by giving the reference a plus or minus sign, respectively [27]. The signs represent the directed arcs in the graph, a positive sign representing the “arrow head” of the arc, and the negative sign the “start” of the arc [9]. Finally, the points that correspond to rows with only positive signs are the topograph minima.

Let us illustrate how the topographical heuristic works by a simple illustrative example, adapted from Ali [27]. Suppose we want to minimize the function

$$f(x, y) = x^2 + y^2, \tag{4}$$

and that six points were sampled and their function values calculated: $f(P_1) = f(2, 5) = 29$, $f(P_2) = f(1, 2) = 5$, $f(P_3) = f(3, 4) = 25$, $f(P_4) = f(0, 1) = 1$, $f(P_5) = f(5, 0) = 25$, and $f(P_6) = f(4, 2) = 20$.

First, the symmetric squared distance matrix \mathbf{D} is constructed, where, for example, the element d_{13} corresponds to the distance between P_1 and P_3 :

$$\mathbf{D} = \begin{bmatrix} 0 & 10 & 2 & 20 & 34 & 13 \\ 10 & 0 & 8 & 2 & 20 & 9 \\ 2 & 8 & 0 & 18 & 20 & 5 \\ 20 & 2 & 18 & 0 & 26 & 17 \\ 34 & 20 & 20 & 26 & 0 & 5 \\ 13 & 9 & 5 & 17 & 5 & 0 \end{bmatrix} \tag{5}$$

Following that, the kNN -matrix is formed by each point’s k -nearest neighbors. Using $k = 3$, the nearest neighbors of P_1 (the first row of \mathbf{D}) are the points with indexes 3, 2, and 6, respectively. These elements will constitute the first row of the matrix. The process goes on until the following matrix is obtained:

$$\mathbf{kNN} = \begin{bmatrix} 3 & 2 & 6 \\ 4 & 3 & 6 \\ 1 & 6 & 2 \\ 2 & 6 & 3 \\ 6 & 2 & 3 \\ 3 & 5 & 2 \end{bmatrix} \tag{6}$$

This matrix represents an undirected graph. Computationally, it is obtained sorting each row of \mathbf{D} and taking the first k elements’ indexes. The elements of the main diagonal of \mathbf{D} receive a very large value (e.g., 10^8) before sorting, so that they are not included in the kNN -matrix.

Now, the elements of kNN will receive a plus or minus sign according to their functional values in relation to the value of the point represented by the row index. The second row, for example, corresponds to P_2 , whose function value is equal to 5, which is more than $f(P_4) = 1$ (P_4 is element knn_{21}), but less than $f(P_3) = 13$ and $f(P_1) = 29$ (elements knn_{22} and knn_{23} , respectively). Therefore, knn_{21} will receive a minus sign and the other two elements a plus sign. The signed matrix becomes

$$kNN = \begin{bmatrix} -3 & -2 & -6 \\ -4 & +3 & +6 \\ +1 & -6 & -2 \\ +2 & +6 & +3 \\ -6 & -2 & +3 \\ +3 & +5 & -2 \end{bmatrix} \quad (7)$$

As the only point that corresponds to a row with only positive signs is $P_4 = (0, 1)$, this will be the starting point for a local optimization algorithm. When implementing the topographical heuristic, the signs can be attributed in the process of construction of kNN .

In step 3, Törn and Viitanen [9] say that any local optimization method can be used. They employed a gradient-based algorithm, as their tests were performed on algebraic test functions. We employ the Hooke–Jeeves algorithm, which is described in the following subsection.

3.2 The Method of Hooke and Jeeves

The Hooke–Jeeves algorithm is a direct search algorithm, as it doesn't make use of derivative information [32]. It performs two types of search: an exploratory search and a pattern search. A summary of the method [33] is given below.

Initialization Step

Let $\mathbf{d}_1, \dots, \mathbf{d}_n$ be the coordinate directions. Choose a scalar $\varepsilon > 0$ to be used for terminating the algorithm. Furthermore, choose an initial step size, $\Delta \geq \varepsilon$, and an acceleration factor, $\alpha > 0$. Choose a starting point, \mathbf{x}_1 , let $\mathbf{y}_1 = \mathbf{x}_1$, let $k = j = 1$, and go to the main step.

Main Step

1. If $f(\mathbf{y}_j + \Delta \mathbf{d}_j) < f(\mathbf{y}_j)$, the trial is termed a *success*; let $\mathbf{y}_{j+1} = \mathbf{y}_j + \Delta \mathbf{d}_j$, and go to Step 2. If, however, $f(\mathbf{y}_j + \Delta \mathbf{d}_j) \geq f(\mathbf{y}_j)$, the trial is deemed a *failure*. In this case, if $f(\mathbf{y}_j - \Delta \mathbf{d}_j) < f(\mathbf{y}_j)$, let $\mathbf{y}_{j+1} = \mathbf{y}_j - \Delta \mathbf{d}_j$, and go to Step 2; if $f(\mathbf{y}_j - \Delta \mathbf{d}_j) \geq f(\mathbf{y}_j)$, let $\mathbf{y}_{j+1} = \mathbf{y}_j$.
2. If $j < n$, replace j by $j + 1$, and repeat Step 1. Otherwise, go to Step 3 if $f(\mathbf{y}_{n+1}) < f(\mathbf{x}_k)$, and go to Step 4 if $f(\mathbf{y}_{n+1}) \geq f(\mathbf{x}_k)$.
3. Let $\mathbf{x}_{k+1} = \mathbf{y}_{n+1}$, and let $\mathbf{y}_1 = \mathbf{x}_{k+1} + \alpha(\mathbf{x}_{k+1} - \mathbf{x}_k)$. Replace k by $k + 1$, let $j = 1$, and go to Step 1.
4. If $\Delta \leq \varepsilon$, stop; \mathbf{x}_k is the solution. Otherwise, replace Δ by $\Delta/2$. Let $\mathbf{y}_1 = \mathbf{x}_k$, $\mathbf{x}_{k+1} = \mathbf{x}_k$, replace k by $k + 1$, let $j = 1$, and repeat Step 1.

Steps 1 and 2 above describe an exploratory search. In step 3, there is an acceleration along the direction $\mathbf{x}_{k+1} - \mathbf{x}_k$. Finally, in step 4 the step size Δ is reduced.

3.3 Description of the new method

The principle behind the new method is quite simple. Firstly, after the crossover process, the individuals are clustered using the topographical heuristic with the k -nearest neighbors. Then, the topograph minima are starting points for instances of the Hooke–Jeeves local search algorithm. If there is improvement, the solution obtained by the local search replaces its correspondent topograph minimum. After that, the standard selection phase occurs. Algorithm 6 displays the method.

4 NUMERICAL COMPARISONS

4.1 Implementation and setup

Our tests were performed on a PC with 8 Gb RAM running Windows10 with a Intel Core i5-2410M processor. Our optimization method was implemented in C using CodeBlock as a compiler. For its stochastic part, we used the pseudorandom number generating algorithm developed by Matsumoto and Nishimura [34].

The Hooke–Jeeves routine inside our method was set up, for all the executions, with initial step size $\Delta = 10^{-3}$, scalar $\varepsilon = 10^{-3}$ and acceleration factor $\alpha = 0.8$.

Data: NP, CR, F, k .

Result: Optimal solution \mathbf{x}^* , $f(\mathbf{x}^*)$.

begin

$NIter \leftarrow 0$

$f(\mathbf{x}_i^*) \leftarrow 1.0E6$

initialize()

repeat

$NIter \leftarrow NIter + 1$

mutate()

crossover()

topographical()

HookeJeeves()

select()

until a termination criterion is satisfied

Algorithm 2: The new method, TopoDE.

As all optimization problems attacked in this work have known global minima, the new DE variant was run using the same termination criterion as in Siarry et al. [35], Hirsch et al. [36], Rios–Coelho et al. [37], which is ideal for an algorithm’s performance assessment:

$$|f(\mathbf{x}^*) - f(\mathbf{x})| \leq \varepsilon_1 |f(\mathbf{x}^*)| + \varepsilon_2, \quad (8)$$

where $f(\mathbf{x}^*)$ is the global optimum, $f(\mathbf{x})$ is the current best, coefficient $\varepsilon_1 = 10^{-4}$ corresponds to the relative error and $\varepsilon_2 = 10^{-6}$ corresponds to the absolute error [35].

4.2 Problems

4.2.1 Global Optimization Test Functions

As in Hirsch et al. [36], Csendes et al. [38], Hirsch et al. [39], Rios–Coelho et al. [37], we tested our algorithm using a testbed proposed by Hedar and Fukushima [40]. As justified by these authors, “the characteristics of these test functions are diverse enough to cover many kinds of difficulties that arise in global optimization problems”. Indeed, this testbed, whose functions are described in Appendix A, contains functions with multiple global optima, with an isolated global optimum, and with high dimensionality.

4.2.2 Chemical equilibrium problem

These nonlinear systems, introduced by Meintjes and Morgan [41], have been widely employed in the literature [36, 37, 42–44], among others. They concern the combustion of propane (C_3H_8) in air (O_2 and N_2) to form ten products. This chemical reaction generates a system of ten equations in ten unknowns, which can be reduced to a system of five equations in five unknowns [41]. We solve both systems formulating them as optimization problems. To see how this formulation is made, the interested reader should see Appendix B.

First, let’s show the simplest system, consisting of five equations in five unknowns. It’s given by

$$\begin{cases} f_1 = x_1x_2 + x_1 - 3x_5 \\ f_2 = 2x_1x_2 + x_1 + x_2x_3^2 + R_8x_2 - R_5x_5 + 2R_{10}x_2^2 + R_7x_2x_3 + R_9x_2x_4 \\ f_3 = 2x_2x_3^2 + 2R_5x_3^2 - 8x_5 + R_6x_3 + R_7x_2x_3 \\ f_4 = R_9x_2x_4 + 2x_4^2 - 4R_5x_5 \\ f_5 = x_1(x_2 + 1) + R_{10}x_2^2 + x_2x_3^2 + R_8x_2 + R_5x_3^2 + x_4^2 - 1 + R_6x_3 \\ \quad + R_7x_2x_3 + R_9x_2x_4 \end{cases} \quad (9)$$

where

$$\begin{cases} R = 10 \\ R_5 = 0.193 \\ R_6 = 0.002597/\sqrt{40} \\ R_7 = 0.003448/\sqrt{40} \\ R_8 = 0.00001799/40 \\ R_9 = 0.0002155/\sqrt{40} \\ R_{10} = 0.00003846/40 \end{cases}$$

Variables x_i are surrogates for atomic combinations, which means that only positive values make physical sense. Among the four real solutions reported by Meintjes and Morgan [41], only one has all-positive components, $\mathbf{x}^* = (0.0031, 34.59, 0.0650,$

0.8594, 0.0369). Hence, if the search domain is taken from the positive side, as we did using the interval $[0, 100]$, this will be the only solution.

The original system consisting of ten equations in ten unknowns, by its turn, is defined as follows:

$$\begin{cases} f_1 = n_1 + n_4 - 3 \\ f_2 = 2n_1 + n_2 + n_4 + n_7 + n_8 + n_9 + 2n_{10} - R \\ f_3 = 2n_2 + 2n_5 + n_6 + n_7 - 8 \\ f_4 = 2n_3 + n_9 - 4R \\ f_5 = K_5 n_2 n_4 - n_1 n_5 \\ f_6 = K_6 n_2^{1/2} n_4^{1/2} - n_1^{1/2} n_6 \left(\frac{p}{n_T}\right)^{1/2} \\ f_7 = K_7 n_1^{1/2} n_2^{1/2} - n_4^{1/2} n_7 \left(\frac{p}{n_T}\right)^{1/2} \\ f_8 = K_8 n_1 - n_4 n_8 \left(\frac{p}{n_T}\right) \\ f_9 = K_9 n_1 n_3^{1/2} - n_4 n_9 \left(\frac{p}{n_T}\right)^{1/2} \\ f_{10} = K_{10} n_1^2 - n_4^2 n_{10} \left(\frac{p}{n_T}\right) \end{cases} \quad (10)$$

where

$$\begin{cases} n_T = \sum_{i=1}^{10} n_i \\ p = 40 \text{ atm} \\ R = 10 \\ K_5 = 0.193 \\ K_6 = 0.002597 \\ K_7 = 0.003448 \\ K_8 = 0.00001799 \\ K_9 = 0.0002155 \\ K_{10} = 0.00003846 \end{cases}$$

Variables n_i represent the number of moles of product i formed per mole of propane consumed, as given by Table 1.

Table 1: Products of propane combustion [41].

Product	Subscript	Description
CO ₂	1	Carbon dioxide
H ₂ O	2	Water
N ₂	3	Nitrogen
CO	4	Carbon monoxide
H ₂	5	Hydrogen
H	6	Hydrogen atom
OH	7	Hydroxyl radical
O	8	Oxygen atom
NO	9	Nitric oxide
O ₂	10	Oxygen

4.3 Results

Both DE and TOPODE were design to stay in a loop until it reaches the stopping criteria written on equation 8. In order to avoid an infinity loop, in the beginning of each new population, it also checks if the total number of function evaluations (FE) has reached the maximum of 10^6 . In this case the loop stops and this run has failed to find the local minima. It means that successful runs are the ones that stops before the new population reaches 10^6 function evaluations.

Table 2 displays the number of FE necessary for the new method to reach the stopping criterion 8 after one-hundred runs using different seeds for each run. For all of the problems the method achieved 100% success rate, being able to find the local minima in all of executions.

For the sake of comparison, the same functions and systems of equations problems were also solved by using the canonical form of the Differential Evolution Method (DE) and the results are present in Table 3. For some problems, the canonical form of the algorithm was not able to find the global optima reaching, thus, lower success rates.

Despite needing, for some problems, more function evaluations to find the global minima than the canonical DE, TopoDE was successful in all of them. This fact corroborates the efficiency of the new method.

Table 2: TopoDE: average, maximum and minimum numbers of functions evaluations, and success rate.

Problem	Average FE	Maximum FE	Minimum FE	Success Rate
Branin	1,268	4,189	317	100%
Chemical Equilibrium	197,098	1,415,105	20,996	100%
Easom	2,024	5,479	376	100%
Goldstein-Price	1,236	13,407	334	100%
Hartman-3	2,168	7,803	621	100%
Hartman-6	5,710	66,638	1,302	100%
Rosenbrock-2	1,729	12,594	400	100%
Rosenbrock-5	20,213	151,571	1,409	100%
Rosenbrock-10	148,056	1,510,708	4,432	100%
Shekel-5	5,818	19,945	867	100%
Shekel-7	6,974	31,689	876	100%
Shekel-10	6,485	44,180	869	100%
Shubert	1,137	4,615	363	100%
Ten-variable Chemical Equilibrium	3,656,715	133,182,273	2,883	100%
Zakharov-5	4,640	59,207	1,014	100%
Zakharov-10	24,281	252,076	3,790	100%

5 CONCLUSIONS

This paper was written in order to present a new optimization method, the topographical differential evolution (TopoDE), which combines Differential Evolution, the Topographical Algorithm, and Hooke-Jeeves. This algorithm was used, with great success, to find the optima for *FOURTEEN* benchmark test functions and two systems of equations. TopoDE outperformed the canonical form of Differential Evolution, demonstrating, thus, its potential for further application.

ACKNOWLEDGEMENTS

The authors acknowledge the financial support provided by CNPq, Conselho Nacional de Desenvolvimento Científico e Tecnológico, FAPERJ, Fundação Carlos Chagas Filho de Amparo à Pesquisa do Estado do Rio de Janeiro, and CAPES, Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (finance code 001).

REFERENCES

- [1] R. Storn and K. Price. “Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces”. *Journal of Global Optimization*, vol. 11, pp. 341–359, 01 1997.
- [2] S. Das and P. Suganthan. “Differential Evolution: A Survey of the State-of-the-Art”. *Evolutionary Computation, IEEE Transactions on*, vol. 15, pp. 4–31, 03 2011.
- [3] W. F. Sacco, N. Henderson, A. C. Rios-Coelho, M. M. Ali and C. M. N. A. Pereira. “Differential evolution algorithms applied to nuclear reactor core design”. *Annals of Nuclear Energy*, vol. 36, pp. 1093–1099, 08 2009.
- [4] N. Henderson, W. F. Sacco, N. E. Baruffatti and M. Ali. “Calculation of Critical Points of Thermodynamic Mixtures with Differential Evolution Algorithms”. *Industrial & Engineering Chemistry Research - IND ENG CHEM RES*, vol. 49, pp. 1872–1882, 02 2010.
- [5] F. Lobato, V. Steffen, Jr and A. Silva Neto. “Solution of singular optimal control problems using the improved differential evolution algorithm”. *Journal of Artificial Intelligence and Soft Computing Research*, vol. 1, pp. 195–206, 01 2011.
- [6] L. Camps Echevarria, O. Santiago, A. Silva Neto and H. Campos Velho. “An Approach to Fault Diagnosis Using Meta-Heuristics: a New Variant of the Differential Evolution Algorithm”. *Computacion y Sistemas*, vol. 18, pp. 5–17, 03 2014.
- [7] R. Hooke and T. A. Jeeves. “Direct Search Solution of Numerical and Statistical Problem”. *J. ACM*, vol. 8, pp. 212–229, 04 1961.

Table 3: DE: average, maximum and minimum numbers of functions evaluations, and success rate.

Problem	Average FE	Maximum FE	Minimum FE	Success Rate
Branin	720	1,119	422	100%
Chemical Equilibrium	7,080	8,781	5,778	7%
Easom	949	1,393	679	100%
Goldstein-Price	525	525	525	1%
Hartman-3	646	889	341	100%
Hartman-6	4,164	6,941	3,088	69%
Rosenbrock-2	979	1,393	526	73%
Rosenbrock-5	9,448	25,438	5,987	8%
Rosenbrock-10	129,412	260,890	70,074	100%
Shekel-5	2,731	3,487	2,014	96%
Shekel-7	2,531	3,260	1,914	98%
Shekel-10	2,534	3,354	1,911	97%
Shubert	1,620	2,926	863	98%
Ten-variable Chemical Equilibrium	32,899	39,242	27,123	100%
Zakharov-5	4,585	5,184	4,122	100%
Zakharov-10	22,103	23,904	20,143	100%

- [8] V. S. Törn, A. “Topographical Global Optimization”. In *C.A. Floudas and P.M. Pardalos (Eds.), Recent Advances in Global Optimization*, pp. 384–398. Princeton University Press, 1992.
- [9] V. S. Törn, A. “An Enhanced Memetic Differential Evolution in Filter Design for Defect Detection in Paper Production”. *Journal of Global Optimization*, vol. 5, pp. 267–276, 1994.
- [10] V. S. Törn, A. “Iterative Topographical Global Optimization”. In *C.A. Floudas and P.M. Pardalos (Eds.), State of the Art in Global Optimization*, pp. 353–363. Kluwer Academic Publishers, 1996.
- [11] W. F. Sacco, N. Henderson and A. C. Rios-Coelho. “Topographical global optimization applied to nuclear reactor core design: Some preliminary results”. *Annals of Nuclear Energy*, vol. 65, pp. 166–173, 03 2014.
- [12] N. Henderson, M. de Sa Rego, W. F. Sacco and R. A. Rodrigues Jr. “A new look at the topographical global optimization method and its application to the phase stability analysis of mixtures”. *Chemical Engineering Science*, vol. 127, pp. 151–174, 05 2015.
- [13] M. Ali and A. Torn. “Optimization of Carbon and Silicon Cluster Geometry for Tersoff Potential using Differential Evolution”. pp. 287–300, 01 2000.
- [14] A. Caponio, G. L. Cascella, F. Neri, N. Salvatore and M. Sumner. “A Fast Adaptive Memetic Algorithm for Online and Offline Control Design of PMSM Drives”. *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society*, vol. 37, pp. 28–41, 03 2007.
- [15] J. Nelder and R. Mead. “A Simplex Method for Function Minimization”. *Computer J.*, vol. 7, pp. 308–313, 01 1965.
- [16] V. Tirronen, F. Neri, T. Karkakinen, K. Valjus and T. Rossi. “An Enhanced Memetic Differential Evolution in Filter Design for Defect Detection in Paper Production”. *Evolutionary Computation*, vol. 16, pp. 529–555, 2008.
- [17] I. Moser and R. Chiong. “A Hooke-Jeeves Based Memetic Algorithm for Solving Dynamic Optimisation Problems”. pp. 301–309, 06 2009.
- [18] S. Boettcher and A. G. Percus. “Extremal Optimization: Methods derived from Co-Evolution.” In *Proceedings of Genetic and Evolutionary Computation Conference*, pp. 825–832, 01 1999.
- [19] S. Dominguez and E. Mezura-Montes. “A cost-benefit local search coordination in multimeme differential evolution for constrained numerical optimization problems”. *Swarm and Evolutionary Computation*, vol. 39, pp. 249–266, 2018.
- [20] J. Lampinen and I. Zelinka. “Mixed variable non-linear optimization by differential evolution”. In *2nd International Prediction Conference*, volume 99, pp. 44–55. Proceedings of the Nostradamus, 1999.

- [21] P. Kaelo and M. Ali. “A numerical study of some modified differential evolution algorithms”. *European Journal of Operational Research*, vol. 169, pp. 1176–1184, 03 2006.
- [22] R. W. Becker and G. V. Lago. “A global optimization algorithm”. *Proceedings of the 8th Allerton Conference on Circuits and Systems Theory*, pp. 3–12, 01 1970.
- [23] A. Törn. “Global optimization as a combination of global and local search.” In *Proceedings of Computer Simulation Versus Analytical Solutions for Business and Economic Models*, pp. 191–206, 1973.
- [24] A. A. Törn. “A search-clustering approach to global optimization”. In *L.C.W. Dixon and G.P. Szegő (Eds.), Towards Global Optimization 2*, pp. 49–62. North-Holland Publishing Company, 1978.
- [25] T. Timmer. “Long Short-Term Memory in Recurrent Neural Networks”. Ph.D. thesis, Erasmus University, Rotterdam, The Netherlands, 1984.
- [26] M. Ali and C. Storey. “Topographical Multilevel Single Linkage”. *Journal of Global Optimization*, vol. 5, pp. 349–358, 12 1994.
- [27] M. Ali, C. Storey and A. Torn. “Application of Stochastic Global Optimization Algorithms to Practical Problems”. *Journal of Optimization Theory and Applications*, vol. 95, pp. 545–563, 01 1997.
- [28] A. Levi and S. Haas. “Appendix A - Global optimization algorithms”. *Optimal Device Design*, pp. 262–276, 01 2010.
- [29] A. Törn and A. Zilinskas. “Global Optimization”. In *G. Goos and J. Hartmanis (Eds.), Lecture Notes in Computer Science, No. 350*. Springer-Verlag, 1989.
- [30] J. Gentle. *Random Number Generation and Monte Carlo Methods*. Statistics and Computing. Springer New York, 2013.
- [31] W. Press, S. Teukolsky, W. Vetterling and B. Flannery. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, 2007.
- [32] R. P. Brent. *Algorithms for minimization without derivatives*. Englewood Cliffs, N.J. : Prentice-Hall, 1973.
- [33] H. A. Rosales-Macedo, M. S. Bazaraa, H. Sherali and C. M. Shetty. “Nonlinear Programming: Theory and Algorithms (2nd Edition)”. *The Journal of the Operational Research Society*, vol. 45, pp. 846, 07 1994.
- [34] M. Matsumoto and T. Nishimura. “Mersenne Twister: a 623-dimensionally equidistributed uniform pseudo-random number generator”. *ACM Transactions on Modeling and Computer Simulation*, vol. 8, pp. 3–30, 01 1998.
- [35] P. Siarry, G. Berthiau, F. Durbin and J. Haussy. “Enhanced Simulated Annealing for Globally Minimizing Functions of Many-Continuous Variables”. *ACM Trans. Math. Softw.*, vol. 23, pp. 209–228, 06 1997.
- [36] M. J. Hirsch, C. N. Meneses, P. Pardalos and M. Resende. “Global optimization by continuous GRASP”. *Optimization Letters*, vol. 1, pp. 201–212, 03 2007.
- [37] A. C. Rios-Coelho, W. F. Sacco and N. Henderson. “A Metropolis algorithm combined with Hooke–Jeeves local search method applied to global optimization”. *Applied Mathematics and Computation*, vol. 217, pp. 843–853, 09 2010.
- [38] T. Csendes, L. Pál, J. O. H. Sendín and J. R. Banga. “The GLOBAL optimization method revisited”. *Optimization Letters*, vol. 2, pp. 445–454, 08 2008.
- [39] M. J. Hirsch, P. Pardalos and M. Resende. “Speeding up continuous GRASP.” *European Journal of Operational Research*, vol. 205, pp. 507–521, 01 2010.
- [40] A.-R. Hedar and M. Fukushima. “Tabu Search directed by direct search methods for nonlinear global optimization”. *European Journal of Operational Research*, vol. 170, pp. 329–349, 04 2006.
- [41] K. Meintjes and A. P. Morgan. “Chemical equilibrium system as numerical test problems”. *ACM Trans. Math. Softw.*, vol. 16, pp. 143–151, 06 1990.
- [42] C. Maranas and C. A. Floudas. “All Solutions of Nonlinear Constrained Systems of Equations”. *Journal of Global Optimization*, vol. 7, pp. 143–182, 01 1995.
- [43] P. Van Hentenryck, D. McAllester and D. Kapur. “Solving Polynomial Systems Using a Branch and Prune Approach.” volume 34, pp. 797–827, 1997.
- [44] C. Grosan and A. Abraham. “A New Approach for Solving Nonlinear Equations Systems”. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 38, pp. 698–714, 06 2008.

- [45] L. Dixon and G. Szegö. *Towards global optimisation 2. Towards global optimisation / eds L. C. W. Dixon and G. P. Szegö.* North-Holland Pub. Co., 1978.
- [46] Z. Michalewicz. *Genetic Algorithm+Data Structures=Evolution Programs.* 01 1996.
- [47] A. Levy and A. Montalvo. “The Tunneling Algorithm for the Global Minimization of Functions”. *SIAM Journal on Scientific and Statistical Computing*, vol. 6, no. 1, pp. 15–29, 1985.
- [48] J. J. Moré, B. S. Garbow and K. E. Hillstom. “Testing Unconstrained Optimization Software”. *ACM Transactions on Mathematical Software*, vol. 7, pp. 17–41, 03 1981.

Appendix A Function Definitions

A.1 Branin [45].

$f(\mathbf{x}) = a(x_2 - bx_1^2 + cx_1 - d)^2 + g(1 - h)\cos(x_1) + g$,
 where $a = 1, b = 5/(4\pi^2), c = 5/\pi, d = 6, g = 10, h = 1/(8\pi)$.
 Domain: $[-5, 15]^2$
 Global minima: $\mathbf{x}^* = (-\pi, 12.275), (\pi, 2.275), (3\pi, 2.475); f(\mathbf{x}^*) = 5/(4\pi)$.

A.2 Easom [46].

$f(\mathbf{x}) = -\cos(x_1)\cos(x_2)e^{-(x_1-\pi)^2-(x_2-\pi)^2}$
 Domain: $[-100, 100]^2$
 Global minimum: $\mathbf{x}^* = (\pi, \pi); f(\mathbf{x}^*) = 1$.

A.3 Goldstein–Price [45].

$f(\mathbf{x}) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)]$
 $[30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$
 Domain: $[-2, 2]^2$
 Global minimum: $\mathbf{x}^* = (0, -1); f(\mathbf{x}^*) = 3$.

A.4 Shubert [47].

$f(\mathbf{x}) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)]$
 Domain: $[-10, 10]^2$
 Global minima:
 $\mathbf{x}^* \approx (-7.0835, 4.8580), (-7.0835, -7.7083), (-1.4251, -7.0835), (5.4828, 4.8580),$
 $(-1.4251, -0.8003), (4.8580, 5.4828), (-7.7083, -7.0835), (-7.0835, -1.4251),$
 $(-7.7083, -0.8003), (-7.7083, 5.4828), (-0.8003, -7.7083), (-0.8003, -1.4251),$
 $(-0.8003, 4.8580), (-1.4251, 5.4828), (5.4828, -7.7083), (4.8580, -7.0835),$
 $(5.4828, -1.4251), (4.8580, -0.8003); f(\mathbf{x}^*) \approx -186.7309$.

A.5 Hartmann [45].

$f_{n,m}(\mathbf{x}) = -\sum_{i=1}^m \alpha_i e^{-\sum_{j=1}^n A_{ij}^{(n)}(x_j - P_{ij}^{(n)})^2}$, where

$$\mathbf{A}^{(3)} = \begin{bmatrix} 3.0 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3.0 & 10 & 30 \\ 0.1 & 10 & 35 \end{bmatrix}$$

$$\mathbf{P}^{(3)} = 10^{-4} \begin{bmatrix} 6890 & 1170 & 2673 \\ 4699 & 4387 & 7470 \\ 1091 & 8732 & 5547 \\ 381 & 5743 & 8838 \end{bmatrix}$$

$$\mathbf{A}^{(6)} = \begin{bmatrix} 10 & 3 & 17 & 3.05 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{bmatrix}$$

$$\mathbf{P}^{(6)} = 10^{-4} \begin{bmatrix} 1312 & 1696 & 5569 & 124 & 8283 & 5886 \\ 2329 & 4135 & 8307 & 3736 & 1004 & 9991 \\ 2348 & 1451 & 3522 & 2883 & 3047 & 6650 \\ 4047 & 8828 & 8732 & 5743 & 1091 & 381 \end{bmatrix}$$

$$\alpha = [1, 1.2, 3, 3.2]$$

Domain: $[0, 1]^n$

Global minimum:

$$n = 3, m = 4: \mathbf{x}^* \approx (0.114614, 0.555649, 0.852547); f_{3,4}(\mathbf{x}^*) \approx -3.86278.$$

$$n = 6, m = 4: \mathbf{x}^* \approx (0.201690, 0.150011, 0.476874, 0.275332, 0.311652, 0.657300);$$

$$f_{6,4}(\mathbf{x}^*) \approx -3.32237.$$

A.6 Rosenbrock [48].

$$f(\mathbf{x}) = \sum_{j=1}^{n-1} [100(x_j^2 - x_{j+1})^2 + (x_j - 1)^2]$$

Domain: $[-10, 10]^n$

Global minimum: $\mathbf{x}^* = (1, \dots, 1); f(\mathbf{x}^*) = 0.$

A.7 Shekel [45].

$$f_{4,m}(\mathbf{x}) = - \sum_{i=1}^m [(x - a_i)^T (x - a_i) + c_i]^{-1}, \text{ where}$$

$$\mathbf{a} = \begin{bmatrix} 4.0 & 4.0 & 4.0 & 4.0 \\ 1.0 & 1.0 & 1.0 & 1.0 \\ 8.0 & 8.0 & 8.0 & 8.0 \\ 6.0 & 6.0 & 6.0 & 6.0 \\ 7.0 & 3.0 & 7.0 & 3.0 \\ 2.0 & 9.0 & 2.0 & 9.0 \\ 5.0 & 5.0 & 3.0 & 3.0 \\ 8.0 & 1.0 & 8.0 & 1.0 \\ 6.0 & 2.0 & 6.0 & 2.0 \\ 7.0 & 3.6 & 7.0 & 3.6 \end{bmatrix}$$

$$\mathbf{c} = [0.1 \ 0.2 \ 0.2 \ 0.4 \ 0.4 \ 0.6 \ 0.3 \ 0.7 \ 0.5 \ 0.5]$$

Domain: $[0, 10]^4$

Global minimum: $\mathbf{x}^* = (4, 4, 4, 4);$

$$f_{4,5}(\mathbf{x}^*) \approx -10.1532, f_{4,7}(\mathbf{x}^*) \approx -10.4029, f_{4,10}(\mathbf{x}^*) \approx -10.5364.$$

A.8 Zakharov [40].

$$f(\mathbf{x}) = \sum_{i=1}^n x_i^2 + (\sum_{i=1}^n 0.5ix_i)^2 + (\sum_{i=1}^n 0.5ix_i)^4$$

Domain: $[-5, 10]^n$

Global minimum: $\mathbf{x}^* = (0, \dots, 0); f(\mathbf{x}^*) = 0.$

Appendix B Nonlinear systems formulated as optimization problems

Let us consider the problem of computing solutions of nonlinear systems with simple bound constraints. We can express this problem as

$$\begin{cases} f_1(\mathbf{x}) = 0 \\ f_2(\mathbf{x}) = 0 \\ \vdots \\ f_N(\mathbf{x}) = 0 \end{cases} \text{ s.t. } \mathbf{x} \in [\mathbf{a}, \mathbf{b}] \subseteq \mathfrak{R}^n, \quad (\text{B-1})$$

where $\mathbf{x} = (x_1, \dots, x_N)^T \in \mathfrak{R}^n$, $f_i : \mathfrak{R}^n \rightarrow \mathfrak{R}$ and $[\mathbf{a}, \mathbf{b}] \equiv [a_1, b_1] \times [a_2, b_2] \times \dots \times [a_N, b_N]$, with $a_i < b_i$, for all $i = 1, \dots, N$. Note that vectors $\mathbf{a} = (a_1, a_2, \dots, a_N)$ and $\mathbf{b} = (b_1, b_2, \dots, b_N)$ are specified as the lower and upper bounds of the variables, and set $[\mathbf{a}, \mathbf{b}]$ is a box in \mathfrak{R}^n , where there exist one or more roots of the nonlinear system.

Let us suppose that function $f_i : \mathfrak{R}^n \rightarrow \mathfrak{R}$, for any $i = 1, \dots, N$, can be nondifferentiable or even discontinuous, but it must be bounded in $[\mathbf{a}, \mathbf{b}]$.

If $F = (f_1(\mathbf{x}), \dots, f_N(\mathbf{x}))^T$, the problem described by Eq. (B-1) can be reformulated as the following optimization problem:

$$\text{Min } f(\mathbf{x}) \text{ s.t. } \mathbf{x} \in [\mathbf{a}, \mathbf{b}] \subseteq \mathfrak{R}^n \quad (\text{B-2})$$

In Eq. (B-2), $f : [\mathbf{a}, \mathbf{b}] \subset \mathfrak{R}^n \rightarrow \mathfrak{R}$ is a nonnegative and possibly multimodal merit function, given by

$$f(\mathbf{x}) = F^T(\mathbf{x})F(\mathbf{x}), \quad (\text{B-3})$$

Since the system represented by Eq. (B-1) has solution(s) in $[\mathbf{a}, \mathbf{b}]$, then, in terms of results, to solve this system is equivalent to find the global minimum(a) of the optimization problem given by Eq. (B-2).