Andrei PĂDUREANU, PhD Candidate
E-mail: andrei.i.padureanu@gmail.com
The Bucharest Academy of Economic Studies

AN EFFICIENT STRATEGY WITHOUT DERIVATIVES FOR BOX CONSTRAINED OPTIMIZATION USING ORTHOGONAL DIRECTIONS

Abstract. *In this paper I present an efficient derivative-free line search strategy using adaptive orthogonal directions for locating a minimum of multivariable box constrained functions. The method which can be considered an advanced development of the well known direct search method of Rosenbrock [28] uses an advanced inexact line search sequence to improve convergence speed and numerical accuracy. Scaled trial steps in every dimension of the search space are considered and every iteration the search directions are changed using Palmer's [24] improved orthogonalization procedure. In the numerical evaluation section the algorithm's performance is compared with other search approaches recently evaluated in Hvattum and Glover [15]. The numerical results show that the algorithm is very competitive in terms of convergence speed. The overall balance between convergence speed, reliability and small arithmetic complexity makes the present routine one of the methods of first choice for employment as local search procedure inside hybrid metaheuristic algorithms.*

Key words: *Continuous optimization, derivative-free methods, metaheuristics, local search algorithms.*

JEL Classification: C61, C63, C65

1. Introduction

One of the most important problems in numerical analysis is finding a local minimum of a multivariable function inside a bounded search space:

$$\min_{X \in \mathbb{R}^n} f(x_1, x_2, \cdots, x_{n-1}, x_n) \qquad (1)$$
$$l_i \le x_i \le u_i, i = \overline{1,n}$$

To solve problem (1) two fundamentally different approaches can be employed.

The first and most common approach is to use a gradient based method. The most popular among gradient based methods is the Newton class which requires first and second order derivatives of the objective function. Methods of this type have very fast local convergence rate to a local minimum if fitted with an appropriate trust region scheme or an inexact line search strategy with guaranteed sufficient decrease to ensure its global convergence properties. Though these methods are fast and usually good implementations yield very good results most of the times in some cases they cannot be the methods of choice. Maybe their biggest drawback is the fact that can be used only on functions that are at least twice differentiable. In some cases even when this property is satisfied the derivatives can be hardly approximated numerically or can be very expensive to compute. Another inconvenient is that they can be used only if the function has a reasonable degree of smoothness. Since a comprehensive discussion about this type of methods is not within the scope of this paper in order to see a proper treatment of it the reader is advised to consult Antoniou and Lu [1].

The second approach of problem (1) is the employment of a derivative-free search algorithm. Methods belonging to this approach are heuristically motivated and in order to determine a descent direction use only the ordinal relation between values of the objective function evaluated in different points of the search space. Since they require no derivatives they can be suited for the minimization of non-smooth and highly nonlinear (sometimes multimodal) functions. The easiness of their employment in practice can be also considered an advantage since their coding does not require knowledge of numerical linear algebra. However not requiring knowledge of numerical linear algebra is the only thing these methods have in common since the way they determine a descent direction is specific to every algorithm. The convergence speed of derivative-free methods is much slower than the speed of the Newton family for problems where the latter can be used. Despite their weaknesses not requiring derivatives makes them more flexible and this is the reason of their employment as local search routines inside hybrid metaheuristic algorithms (see *Chelouah* and *Siarry* [5], [6], *Coelho* et al. [7], *Hedar* and *Fukushima* [10], [11], *Mladenovic et al.* [19] and in *Sacco et al.* [25]).

The most popular among the local derivative-free algorithms employed by hybrid metaheuristics are the well known *Nelder-Mead* [23], *Rosenbrock* [28] and *Hooke* and *Jeeves* [16] direct search methods. The reason of their popularity is the easiness they can be computer programmed. However simplicity is not always the right solution since all 3 methods have their limitations. First of all only Rosenbrock's method [24] is somehow theoretically guaranteed to converge to a local stationary point of a function (as being a derivation of Cauchy's method of steepest descent). Secondly the most popular direct search algorithm, the nonlinear simplex of Nelder and Mead in all its variants [4], [20], [23], [32], has proven in *Han* and *Neumann* [10] and in *Hvattum* and *Glover* [15] to perform very poorly for problems with more than 4 variables. Therefore it is not recommended for employment when dealing with larger problems. On the other hand Rosenbrock's and Hooke-Jeeves' methods perform better than Nelder-Mead for larger problems as seen in [15] but require a large number of function evaluations to reach reasonable numerical accuracy and their reliability is somewhat influenced by the chosen length of the initial trial step.

Since most of the search inside hybrid metaheuristics is performed by local search procedures it is understandable that the overall performance of the algorithm is very dependable on the performance of the local subroutine. As a result a derivative-free procedure capable of dealing efficiently with problem (1) mainly when the objective is a high-dimensional function would be of interest of many fields in which optimization plays an important role.

Because of its desirable properties of theoretically guaranteed convergence, Rosenbrock's method for unconstrained optimization was subject of further development. In 1964 in an internal research note of Imperial Chemical Industries [30] Swann mentioned that by performing a more sophisticated inexact line search sequence along each direction of Rosenbrock's method a new procedure is obtained which proves generally superior in practice to both Rosenbrock's method [24], from which it was developed, and Hooke and Jeeves' pattern search method [13]. In 1965 in a comparative analysis made between several top derivative free methods [8], Roger Fletcher noted that the modified version Rosenbrock's method of the Imperial Chemical technical staff proves both simple and efficient and wandered if it may have any advantages over its competitors using conjugate directions (*Powell* [22], *Zangwill* [31]) when the number of variables is increasing. Since the improved method was considered company's intellectual property, to my knowledge no implementation of it was discussed in a public material. Moreover the method's vague description given recently in Lewis et al. [18] and its absence from the comprehensive comparative study between local derivative-free routines made by Hvattum and Glover [15], made me believe that a comprehensive treatment of this development would prove beneficial.

As a result the rest of this paper is organized as follows. In section 2 I present a detailed description of the enhanced algorithm along with my implementation scheme. In section 3 some numerical examples are used to demonstrate algorithms performance compared with the search approaches from [15] and finally in section 4 a short discussion is given regarding the conclusions of this paper and directions for further research. From the beginning it should be noted that I treat the case of box constrained optimization not the unconstrained one treated by Swann [3], [29] and therefore the procedure presented here is different to the original in some aspects.

2. DESCRIPTION OF THE ALGORITHM

Let's reconsider the general box constrained optimization problem (1). Let $X_0 = (x_1 \quad x_2 \quad \cdots \quad x_{n-1} \quad x_n)^T$ be a feasible starting point of problem (1).

Like Rosenbrock's, the present method employs n orthonormal search directions but unlike its predecessor an advanced inexact linear search sequence is carried out once along each direction in turn using the one-dimensional search algorithm of Davies, Swann and Campey [3]. After a linear search has been performed along each search direction, the distances moved in each them d_i are compared with δ_i the trial step lengths in the corresponding direction and if at least one $|d_i| > \delta_i \; i = \overline{1,n}$ the search directions are redefined using Palmer's [21] improved orthogonalization procedure. If all $|d_i| < \delta_i, i = \overline{1,n}$ then the trial step length is multiplied by a demultiplication factor ***K*** and a further linear search is performed along the n old search direction vectors. Convergence of the method is assumed if one of the trial step-lengths δ_i falls under a predefined value ε.

From the functional point of view the present procedure can be divided in 3 stages: the initialization of the search, the linear approximation of the minimum along each search direction and the change of the search vectors by orthogonalization of the walked distances.

The first stage is performed only once as the search process is initiated while stages 2 and 3 are repeated inside an outer loop until one of the stopping criteria is met. Therefore a standard iteration of the algorithm is formed by a linear search sequence and an orthogonalization procedure.

input : X_0, ε, $MAXEVAL$
output: X^*, f^*, $EVAL$

```
begin
    INITIALIZATION
    while EVAL<MAXEVAL || CONVERGENCE=FALSE do
        while EVAL<MAXEVAL || |d_i| ≤ |δ_i|, i = 1,n do
            INEXACT LINEAR SEARCH SEQUENCE
        if CONVERGENCE=FALSE || EVAL<MAXEVAL then
            CHANGE SEARCH DIRECTIONS
```

Figure 1 - Flowchart of the algorithm

STAGE 1: Initialization.

The purpose of the stage is to initiate the search sequence. In order do that it is necessary to evaluate the objective of the starting point X_0 and to initialize the search directions using n mutually orthonormal vectors $\xi_i^0,\ i = \overline{1,n}$.

$$\begin{array}{lcllllll} \xi_1^0 & = & [1 & 0 & \cdots & \cdots & 0 & 0]^T \\ \xi_2^0 & = & [0 & 1 & \cdots & \cdots & 0 & 0]^T \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \xi_{n-1}^0 & = & [0 & 0 & \cdots & \cdots & 1 & 0]^T \\ \xi_n^0 & = & [0 & 0 & \cdots & \cdots & 0 & 1]^T \end{array} \qquad (2)$$

STAGE 2: Linear minimization sequence.

The goal of this stage is to approximate the distance d_i that minimizes

$$f(X_0 + d_i\xi_i^0),\ i = \overline{1,n} \qquad (3)$$

in every direction $\xi_i^0, i = \overline{1,n}$ using an one-dimensional inexact line search algorithm that does not assume a priori knowledge of minimum's bracketing interval.

The one-dimensional algorithm of *Davies, Swann and Campey* as described in Box et al. [3] combines a search method with an approximation method, the first being used to locate the interval which brackets the minimum and the second to estimate the minimum by quadratic interpolation.

Starting from X_0 a trial step δ_i is taken in the corresponding direction ξ_i. If the objective of the trial point is less than the initial function value $f(X_0)$ the step-length is doubled and further movement is made in the direction in which the function is decreasing. This process is repeated until the minimum has been overshot. Then the step-length is halved and smaller step is taken again from the last successful point. This will give four points equally spaced along the direction of the search. To reduce the uncertainty range the point that is furthest from the point having the smallest objective value is rejected and the remaining three points are used to approximate the minimum by quadratic interpolation. After one linear search has been performed for all ξ_i^0, the distance walked in every direction $d_i, i = \overline{1,n}$ is compared with the initial step length in that direction $\delta_i,\ i = \overline{1,n}$. If at least one $|d_i| > \delta_i$ then search directions are changed via orthogonalization, otherwise step-lengts $(\delta_i, i = \overline{1,n})$ are decreased by a demultiplication factor $0 < K < 1$.

```
while EVAL<MAXEVAL || Convergence=False do
    X_0←X_Best; f_0←f_Best;
    while EVAL<MAXEVAL ||(|d_i| ≤ |δ_i|, i = 1,n) || Convergence=False do
        Compute δ_i←δ_%·(u_i-l_i);
        foreach ξ_i^0 do
            δ ← δ_i; p←0; X_{+δ}←X_Best+δ_i·ξ_i^0; X_{-δ}←X_Best−δ_i·ξ_i^0; f_{+δ}←f(X_{+δ});
            EVAL ← EVAL + 1;
            if f_{+δ}<f_0 || X_{+δ} feasible then
                Set p←1
            else
                f_{-δ}←f(X_{-δ});
                if f_{-δ}<f_0||X_{-δ}feasible then
                    p←−1
                else
                    No search sign case;
                    X_trial←X_0+δ·(f_{-δ}−f_{+δ})/(2·(f_{-δ}−2·f_0+f_{+δ}))·ξ_i^0; f_trial←f(X_trial);
                    EVAL ← EVAL + 1;
                    if f_trial < f_Best || X_trial feasible then
                        X_best←X_trial; f_best←f_trial; d_i←δ·(f_{-δ}−f_{+δ})/(2·(f_{-δ}−2·f_0+f_{+δ}))·ξ_i^0;
            if p≠0 then
                repeat
                    X_{n−2}←X_{n−1}; d_i←d_i + p·δ; δ←2·δ; X_n←X_{n−1} + p·δ·ξ_i^0;
                    if X_n feasible then
                        f_n←f(X_n); EVAL←EVAL + 1;
                until (f_n>f_{n−1})||(X_n infeasible)||(EVAL>MAXEVAL);
                if X_n feasible then
                    δ←δ/2; X_m←X_{n−1} + p·δ·ξ_i^0; f_m←f(X_m); EVAL ← EVAL + 1;
                    if f_m ≥ f_{n−1} then
                        X*←X_{n−1} + p·δ·(f_{n−2}−f_m)/(2·(f_{n−2}−2·f_{n−1}+f_m))·ξ_i^0; f*←f(X*);
                        if f* < f_{n−1} then
                            X_{n−1} ← X*; d_i ← d_i + p·δ·(f_{n−2}−f_m)/(2·(f_{n−2}−2·f_{n−1}+f_m)); f_{n−1} ← f*;
                        X_Best←X_{n−1} f_Best ← f_{n−1};
                    else
                        X*←X_m + p·δ·(f_{n−1}−f_n)/(2·(f_{n−1}−2·f_m+f_n))·ξ_i^0; f*←f(X*);
                        EVAL←EVAL + 1;
                        if f* < f_m then
                            X_{n−1}←X*; d_i ← d_i + p·δ·(f_{n−1}−f_n)/(2·(f_{n−1}−2·f_m+f_n)); f_{n−1} ← f*;
                        else
                            X_{n−1}←X_m; d_i ← d_i + p·δ; f_{n−1} ← f_m;
                        X_Best←X_{n−1}; f_Best ← f_{n−1};
```

Figure 2 - Pseudo-code of the linear search sequence

STAGE 3: Change of the search directions $\xi_i, i = \overline{1,n}$ by orthogonalization

The purpose of this stage is to generate a new set of n orthogonal vectors $\xi_1^1, \xi_2^1, \cdots, \xi_n^1$, such that ξ_1^1 lies in the direction of greatest advance from the previous stage (along the line joining the first and the last points in the n-dimensional space).

The stage consists of 3 steps: the computation of the vectors joining the initial and final points by use of different sets of search vectors (4), the total distance moved in the corresponding sets of directions (5) and the change of search by Palmer's [21] improved orthogonalization procedure (6).

$$\begin{aligned} A_1 &= d_1\xi_1^0 + d_2\xi_2^0 + \cdots\cdots + d_{n-1}\xi_{n-1}^0 + d_n\xi_n^0 \\ A_2 &= d_2\xi_2^0 + \cdots\cdots + d_{n-1}\xi_{n-1}^0 + d_n\xi_n^0 \\ &\cdots\cdots\cdots\cdots\cdots\cdots \\ A_{n-1} &= d_{n-1}\xi_{n-1}^0 + d_n\xi_n^0 \\ A_n &= d_n\xi_n^0 \end{aligned} \tag{4}$$

$$|A_k| = \sqrt{\sum_{i=k}^{n} d_i^2}\ , k = \overline{1,n} \tag{5}$$

$$\xi_k^1 = \frac{d_k A_k - \xi_{k-1}^0 |A_k|^2}{|A_{k-1}| \cdot |A_k|} \quad if\ |A_{k-1}| \cdot |A_k| \neq 0, k = \overline{2,n} \qquad \xi_1^1 = \frac{A_1}{|A_1|} \tag{6}$$

After the orthogonalization process has been finished, the search process is resumed from the inexact line search sequence until one of the termination criteria has been met.

for $i \leftarrow 1$ **to** n **do**
  $A_{i,n} \leftarrow d_n \cdot \xi_0^n$
for $j \leftarrow n-1$ **to** 1 **do**
  for $i \leftarrow 1$ **to** n **do**
    $A_{i,j} \leftarrow A_{i,j+1} + d_j \cdot \xi_0^j$
$|A_n| \leftarrow d_n^2$;
for $i \leftarrow n-1$ **to** 1 **do**
  $|A_i| \leftarrow |A_{i+1}| + d_i^2$
for $i \leftarrow n$ **to** 2 **do**
  if $\sqrt{|A_i| \cdot |A_{i-1}|} \neq 0$ **then**
    $\xi_1^i \leftarrow \frac{d_{i-1} \cdot A_{i,j} - \xi_0^{i-1} \cdot |A_i|}{\sqrt{|A_i| \cdot |A_{i-1}|}}$
$\xi_1^1 \leftarrow \frac{A_{1,1}}{\sqrt{|A_1|}}$

Figure 3 – Pseudo-code of the direction change sequence

3. NUMERICAL EVALUATION

The purpose of this section is to demonstrate the performance of the method described in the previous section by means of comparison with the other derivative-free search approaches recently evaluated [15]. For that reason 8 of the test functions from there (see appendix for details) were used as benchmarks. Like in Hvattum and Glover [15] the functions were shifted such that their global minimum is 0. Also the stopping criteria and initial trial step-lengths used here were the same:

- $f_\varepsilon^* = 0.001,\ f_{Best} < f_\varepsilon^*$
- $EVAL > MAXEVAL = 50\,000$
- $\delta_i = 5\% \cdot (u_i - l_i), i = \overline{1,n}$

Therefore the minimization process has been stopped if either the current best solution had dropped below 0.001 or the function evaluations counter had exceeded 50 000. In the tables containing comparisons of the results, the numbers between parentheses represent the probabilities of reaching $\varepsilon - \text{optimal solutions}$ within the permitted number of function evaluations where failed attempts have been recorded. For every function I used a sample of 50 randomly generated starting points inside the definition box $[l_i, u_i],\ i = \overline{1,n}$ and two settings for the algorithm's step demultiplication factor $\boldsymbol{K}$.

Table 1 - Description of the search mechanisms

Abbreviation	***Method's Name***	***Classification***	***References***
NM	*Nelder-Mead*	*Nonlinear Simplex Search*	[20,32]
MDS	*Multi-Directional Search*	*Nonlinear Simplex Search/Pattern Search*	[30]
CS	*Compass Search*	*Pattern Search/Generating Set Algorithm*	[16]
HJ	*Hooke and Jeeves*	*Pattern Search*	[13]
ROS	*Rosenbrock*	*Direct Search Method with Adaptive Search Directions*	[24]
SW	*Solis and Wets*	*Stochastic Direct Search Method*	[26]
HPS	*Heuristic Pattern Search*	*Derivative-Free Method (not a direct search)*	[11]
SPSA	*Simultaneous Perturbation Stochastic Approximation*	*Derivative-Free Method (not a direct search)*	[27]
SSR	*Scatter Search-Random*	*Stochastic Direct Search with Randomized Subset Generation Algorithm*	[15]
SSC	*Scatter Search-Clustered*	*Stochastic Direct Search with Clustering Subset Generation Algorithm*	[15]
EDSC1	***Present having K=0.2***	***Inexact Line Search with Adaptive Search Directions***	
EDSC2	***Present having K=0.1***		

The overall evaluation results of the present method were quite surprising. It was to be expected that it would prove better than older methods like Rosenbrock's and Hooke and Jeeves' but it managed to outperform all the other search approaches in minimizing 7 out of 8 of the test functions. For every instance of the test the numbers having no parentheses represent the average number of function evaluations obtained by every method during the corresponding test instance. ***n*** denotes the number of variables for every instance of the test problems used for evaluation.

Table 2 - Summary of the test results

Test	*Test function*	*Winner*	*Runner-Up*	*Comments*
1	***Rosenbrock***	***EDSC1,EDSC2***	***ROS, SSR, SSC***	*EDSC methods win for all n except n=64*
2	***Zakharov***	***EDSC1,EDSC2***	***ROS***	*EDSC methods win for all dimensions*
3	***Matyas***	***EDSC1,EDSC2***	***SSC, SSR***	*EDSC methods win for all dimensions*
4	***Sphere***	***EDSC1,EDSC2***	***SSR, SSC***	*EDSC methods win for all dimensions*
5	***Sum Squares***	***EDSC1,EDSC2***	***SSR, SSC***	*EDSC methods win for all dimensions*
6	***Trid***	***EDSC1,EDSC2***	***ROS***	*EDSC methods win for all dimensions*
7	***Booth***	***EDSC1,EDSC2***	***SSC, SSR***	*EDSC methods win for all dimensions*
8	***Branin***	***SSC,SSR***	***EDSC1, EDSC2***	*EDSC methods place* 2^{nd}

Due to the accelerating effect of the quadratic interpolation step, the present method shows remarkable convergence properties when functions are convex or have wide convex neighborhoods. When minimizing Zakharov's function the present method is the only one capable to reach $\varepsilon - \text{optimal values}$ within 50.000 evaluations for the 128 variable case and managed that using 3 times less evaluations than the runner-up, the classic Rosenbrock method, needed in order to find one in the 64 variable case.

For the 512 variable version of the Matyas function only the variants of the present method found ε optimal solutions within the allowed number of evaluations. Also, for test problems 4 and 5 the difference between the present method and its followers is substantial, ***EDSC*** being at least 70 % faster in terms evaluations. Another remarkable result is obtained for test problem number 6, ***EDSC*** being the only method which manages to find $\varepsilon -$ optimal solutions for the 64 variable case. Last but not least, for the Branin test function, ***EDSC*** finishes second but is the only routine except the winners able to find $\varepsilon -$ optimal solutions for all sizes of the problem used in the test.

Table 3 - Comparative results in minimizing Rosenbrock function

n	*NM*	*MDS*	*CS*	*HJ*	*ROS*	*SW*	*HPS*	*SPSA*	*SSR*	*SSC*	*EDSC1*	*EDSC2*
2	466.4	(0.8)	17083.2	(0.7)	237.8	9434.1	(0.0)	(0.0)	696.5	313.2	**179.12**	**194.58**
4	2290.3	(0.0)	(0.8)	(0.4)	1125.0	25580.4	(0.1)	(0.0)	13410.0	2256.4	**431.13**	**464.74**
8	(0.0)	(0.0)	(0.4)	(0.0)	2761.0	(0.0)	(0.0)	(0.0)	25831.6	8165.6	**1317.87**	**1504.67**
16	(0.0)	(0.0)	(0.0)	(0.0)	7820.2	(0.0)	(0.0)	(0.0)	(0.9)	12500.4	**4801.38**	**5087**
32	(0.0)	(0.0)	(0.0)	(0.0)	21183.1	(0.0)	(0.0)	(0.0)	(0.9)	21397.7	**18259.2**	**20678.9**
64	(0.0)	(0.0)	(0.0)	(0.0)	(0.1)	(0.0)	(0.0)	(0.0)	39193.2	(0.2)	**(0.0)**	**(0.0)**
128	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	**(0.0)**	**(0.0)**

Table 4 - Comparative results in minimizing Zakharov function

n	*NM*	*MDS*	*CS*	*HJ*	*ROS*	*SW*	*HPS*	*SPSA*	*SSR*	*SSC*	*EDSC1*	*EDSC2*
2	53.4	43.8	46.9	69.8	35.9	70.7	87.3	779.7	49.2	49.3	**22.4**	**22.9**
4	339.1	161.0	174.9	260.0	134.3	161.7	150.7	5867.0	168.1	173.2	**56.3**	**57.8**
8	(0.0)	1037.7	1535.9	1930.1	477.6	630.8	7050.5	(0.3)	1060.9	1101.1	**171.4**	**179.5**
16	(0.0)	10872.5	21579.7	26317.6	1784.2	3217.7	(0.7)	(0.0)	6827.8	4030.5	**613.9**	**652.2**
32	(0.0)	(0.0)	(0.0)	(0.0)	7696.0	21876.6	(0.0)	(0.0)	32626.4	16167.1	**1982.2**	**2040.1**
64	(0.0)	(0.0)	(0.0)	(0.0)	28022.2	(0.0)	(0.0)	(0.0)	(0.0)	(0.1)	**7021.5**	**7180.8**
128	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	**27825.0**	**29431.3**

Table 5 - Comparative results in minimizing Matyas function

n	*NM*	*MDS*	*CS*	*HJ*	*ROS*	*SW*	*HPS*	*SPSA*	*SSR*	*SSC*	*EDSC1*	*EDSC2*
2	35.5	221.4	57.2	71.4	46.2	59.7	98.1	(0.1)	60.3	47.6	**39.4**	**47.1**
4	282.5	999.4	185.4	236.6	141.5	240.3	316.8	(0.0)	209.9	167.2	**131.6**	**139.8**
8	(0.0)	5031.4	479.3	659.7	344.0	582.4	791.0	(0.0)	432.4	380.0	**354.4**	**374.2**
16	(0.0)	21069.7	1273.0	1816.3	899.5	1589.0	2503.4	(0.0)	1109.8	902.1	**832.1**	**857.3**
32	(0.0)	(0.0)	2926.0	4376.6	2356.6	3614.0	5329.8	(0.0)	2422.8	1834.6	**1997.6**	**2094.9**
64	(0.0)	(0.0)	6961.2	9928.1	6351.4	8433.8	20274.1	(0.0)	3854.2	3827.1	**4444.8**	**4623.5**
128	(0.0)	(0.0)	15601.1	22486.5	30923.4	18173.9	(0.0)	(0.0)	8575.0	9687.9	**9922.1**	**10287.4**
256	(0.0)	(0.0)	35836.5	(0.5)	(0.0)	39372.8	(0.0)	(0.0)	24207.0	24903.1	**21388.2**	**22167.9**
512	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	**46237.1**	**47061.4**

Table 6 - Comparative results in minimizing Sphere function

n	*NM*	*MDS*	*CS*	*HJ*	*ROS*	*SW*	*HPS*	*SPSA*	*SSR*	*SSC*	*EDSC1*	*EDSC2*
2	67.2	31.0	36.3	52.3	25.3	55.3	65.2	767.1	39.4	34.0	**10.8**	**10.8**
4	283.2	126.6	95.8	129.5	64.9	89.0	104.1	901.1	63.8	63.3	**20.7**	**20.7**
8	(0.9)	524.2	223.4	284.3	156.0	172.3	177.9	1092.0	114.7	120.6	**40.4**	**40.4**
16	(0.2)	2385.0	618.4	618.4	371.8	373.5	406.3	1277.3	227.2	232.8	**80.8**	**80.8**
32	(0.0)	10535.4	1107.2	1256.3	1082.8	784.8	1062.2	1652.4	462.5	463.2	**159.5**	**159.5**
64	(0.0)	46657.0	2440.1	2784.0	(0.9)	1602.2	3847.2	2536.7	958.0	983.9	**318.3**	**318.3**
128	(0.1)	(0.0)	5128.7	5642.5	(0.7)	3515.3	18226.9	5170.5	1979.4	2111.7	**635.0**	**635.0**
256	(0.0)	(0.0)	11113.2	12565.9	(0.3)	7401.3	(0.0)	(0.0)	4236.8	4258.6	**1267.1**	**1267.1**
512	(0.0)	(0.0)	23070.3	25148.6	(0.0)	15770.9	(0.0)	(0.0)	8890.4	8890.2	**2536.1**	**2536.1**

Table 7 - Comparative results in minimizing Sum Squares function

n	*NM*	*MDS*	*CS*	*HJ*	*ROS*	*SW*	*HPS*	*SPSA*	*SSR*	*SSC*	*EDSC1*	*EDSC2*
2	42.7	39.0	43.1	61.6	33.3	63.5	88.7	(846.6)	41.4	40.4	**10.9**	**10.9**
4	405.9	169.8	119.3	157.7	102.5	117.4	130.8	(832.1)	77.7	81.2	**20.8**	**20.8**
8	(0.4)	692.2	292.6	362.6	247.6	347.5	391.5	(846.6)	146.3	146.2	**40.3**	**40.3**
16	(0.0)	3255.4	683.8	798.1	743.1	1262.9	(0.4)	(2463.8)	298.3	313.0	**80.8**	**80.8**
32	(0.0)	15233.0	1543.3	1746.3	2507.4	4751.7	(0.2)	(11616.7)	622.5	647.5	**160.2**	**160.2**
64	(0.0)	(0.0)	3457.0	3791.5	15871.6	18346.8	(0.0)	(0.0)	1369.0	1348.4	**317.3**	**317.3**
128	(0.1)	(0.0)	7572.9	8245.4	(0.0)	(0.0)	(0.3)	(0.0)	2840.7	2808.8	**634.7**	**634.7**
256	(0.0)	(0.0)	16411.9	17771.5	(0.0)	(0.0)	(0.0)	(0.0)	6057.1	6105.4	**1267.9**	**1267.9**
512	(0.0)	(0.0)	35319.0	38177.1	(0.0)	(0.0)	(0.0)	(0.0)	12866.2	13021.4	**2533.4**	**2533.4**

Table 8 - Comparative results in minimizing Trid function

n	*NM*	*MDS*	*CS*	*HJ*	*ROS*	*SW*	*HPS*	*SPSA*	*SSR*	*SSC*	*EDSC1*	*EDSC2*
2	36.0	72.0	40.3	53.5	39.0	54.0	88.0	3005.7	39.5	43.0	**31.9**	**33.3**
4	750.5	969.8	172.2	247.9	165.2	178.2	211.6	32171.9	166.0	152.7	**113.4**	**120.2**
8	(0.0)	14409.0	1230.4	1498.0	646.0	1802.1	5633.0	(0.0)	676.3	563.5	**437.9**	**450.8**
16	(0.0)	(0.0)	8811.5	8981.0	3041.8	16053.8	(0.6)	(0.0)	4357.8	3447.5	**1799.3**	**1839.4**
32	(0.0)	(0.0)	(0.0)	(0.0)	14499.4	(0.0)	(0.0)	(0.0)	(0.8)	(0.3)	**7961.5**	**8613.8**
64	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	**37627.0**	**38673.9**

Table 9 - Comparative results in minimizing Booth function

n	*NM*	*MDS*	*CS*	*HJ*	*ROS*	*SW*	*HPS*	*SPSA*	*SSR*	*SSC*	*EDSC1*	*EDSC2*
2	35.2	56.4	71.4	106.9	51.3	75.4	103.8	957.9	63.0	70.2	**43.0**	**46.9**
4	316.3	177.6	189.5	274.6	150.7	189.8	172.8	1183.1	203.1	192.7	**124.2**	**125.6**
8	(0.3)	760.3	491.3	640.7	387.0	510.9	525.2	1701.6	422.8	413.0	**276.4**	**295.9**
16	(0.0)	3217.2	1163.3	1505.7	915.3	1284.6	(0.9)	2660.8	809.4	737.2	**587.1**	**615.3**
32	(0.0)	13946.4	2643.8	2872.2	2512.3	2878.5	(0.7)	5373.6	1732.4	1436.9	**1269.1**	**1303.5**
64	(0.0)	(0.0)	6040.0	6304.9	9873.0	6148.3	(0.7)	13806.5	3154.8	3148.5	**2619.3**	**2649.3**
128	(0.0)	(0.0)	13403.9	16967.0	(0.8)	13015.3	(0.5)	41173.0	6737.8	7516.3	**5549.7**	**5559.0**
256	(0.0)	(0.0)	29754.9	41187.0	(0.0)	27799.6	(0.0)	(0.0)	16293.8	16915.3	**11962.1**	**12050.9**
512	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	36840.3	38635.4	**26118.1**	**26231.0**

Table 10 - Comparative results in minimizing Branin function

n	*NM*	*MDS*	*CS*	*HJ*	*ROS*	*SW*	*HPS*	*SPSA*	*SSR*	*SSC*	*EDSC1*	*EDSC2*
2	43.8	57.2	51.6	67.9	44.3	71.1	88.6	1142.7	57.1	70.2	**38.2**	**39.3**
4	224.0	188.4	116.0	171.5	116.6	143.5	155.2	1406.9	102.8	192.7	**105.4**	**120.6**
8	(0.0)	663.7	290.5	394.8	282.9	341.1	272.9	1538.4	197.1	413.0	**250.2**	**266.9**
16	(0.0)	3015.7	680.5	979.9	647.8	928.9	934.9	1926.5	406.6	737.2	**595.2**	**647.2**
32	(0.0)	11516.6	1550.7	(0.9)	1542.2	2032.8	(0.8)	2819.7	779.7	1436.9	**1292.8**	**1363.6**
64	(0.0)	46751.7	3628.4	(0.9)	6544.6	4558.6	5664.5	(0.7)	1636.6	3148.5	**2682.2**	**2824.4**
128	(0.0)	(0.0)	8031.5	(0.8)	28666.7	9922.3	21469.0	(0.0)	3448.8	7516.3	**5730.8**	**5986.4**
256	(0.0)	(0.0)	18172.5	(0.6)	(0.1)	24043.6	(0.0)	(0.0)	7173.4	16915.3	**12918.2**	**13181.4**
512	(0.0)	(0.0)	(0.9)	(0.7)	(0.0)	(0.2)	(0.0)	(0.0)	15206.9	38635.4	**28418.4**	**28600.6**

4. CONCLUSIONS AND FUTURE RESEARCH

In this paper an efficient derivative-free line search strategy using adaptive orthogonal search directions for local box constrained optimization was presented along with its detailed implementation scheme.

The method was thoroughly evaluated using well known benchmark functions and compared with other methods of its kind. The evaluation results prove that the presented routine is a very tough contender for the derivative-free search methods evaluated in Hvattum and Glover [15].

The quadratic interpolation step of the linear search stage has a very powerful accelerating effect on the convergence speed especially on neighborhoods where the objective function is convex. This is particularly useful when dealing with convex functions or with functions having wide convex ranges as it was observed in the cases of Zakharov, Matyas, Sphere, Sum Squares and Trid functions.

The algorithm's overall balance between convergence speed, simplicity and small arithmetic complexity makes it one of the methods of first choice for employment as local search subroutine inside metaheuristic hybrids.

The detailed description of the multidimensional linear search stage should prove useful in the implementation of other derivative free methods such as conjugate direction methods of Powell [22] and Zangwill [31].

I believe that implementations of EDSC using other one-dimensional minimization procedures for performing linear search stage and a comparison with the present one might lead to interesting findings.

Future work consisting in implementations of EDSC as local search subroutine inside a metaheuristic hybrid such as generalized variable neighborhood search [19], [9] or as tabu search [5], [9] would also confirm the method's potential.

ACKNOWLEDGEMENT

This article is a result of the project POSDRU/88/1.5./S/55287 "Doctoral Programme in Economics at European Knowledge Standards (DOESEC)". This project is co-funded by the European Social Fund through The Sectorial Operational Programme for Human Resources Development 2007-2013, coordinated by The Bucharest Academy of Economic Studies in partnership with West University of Timisoara.

REFERENCES

[1]**Antoniou A., Lu W.-S. (2007)**, *Practical Optimization: Algorithms and Engineering Applications*, Springer, p.101-106;

[2]**Alkhamis T.M., Mohamed A.A. (2006)**, *A Modified Hooke and Jeeves Algorithm with Likelihood Ratio Performance Extrapolation for Simulation Optimization*, European Journal of Operational Research, 174, p.1802–1815;

[3]**Box M.J., Davies D., Swann W.H. (1969)**, ***Non-Linear Optimization Techniques,*** Imperial Chemical Industries *Monograph No.5*, Oliver & Boyd, Edinburgh, p.14-15,27-29;
[4]**Burmen A., Puhan J., Tuma T. (2006),** ***Grid Restrained Nelder-Mead Algorithm,*** Computational Optimization and Applications, 34, p.359–375;
[5]**Chelouah R., Siarry P. (2005)**, ***A Hybrid Method Combining Continuous Tabu Search and Nelder–Mead Simplex Algorithms for the Global Optimization of Multiminima Functions,*** European Journal of Operational Research, 161, p.636–654;
[6]**Chelouah R., Siarry P. (2003)**, ***Genetic and Nelder–Mead Algorithms Hybridized for a More Accurate Global Optimization of Continuous Multiminima Functions***, European Journal of Operational Research, 148, p.335–348;
[7]**Coelho A.C., Sacco W.F., Henderson N. (2010)**, ***A Metropolis Algorithm Combined with Hooke–Jeeves Local Search Method Applied to Global Optimization,*** Applied Mathematics and Computation, 217, p.843-853;
[8]**Fletcher R. (1965)**, ***Function Minimization without Evaluating Derivatives—A Review,*** The Computer Journal, 8, p.33-41;
[9]**Glover F., Kochenberger G.A. (2010)**, ***Handbook of Methaheuristics***, second edition, Preface to First Edition, Springer, p.ix-xi.;
[10]**Han L., Neumann M. (2006)**, ***Effect of dimensionality on the Nelder–Mead simplex method,*** Optimization Methods and Software, 21, p.1–16.
[11]**Hedar A., Fukushima M. (2004)**, ***Heuristic Pattern Search and its Hybridization with Simulated Annealing for Nonlinear Global Optimization,*** Optimization Methods and Software, 19, p.291–308;
[12]**Hedar A., Fukushima M. (2006)**, ***Tabu Search Directed by Direct Search Methods for Nonlinear Global Optimization***, European Journal of Operational Research, 170, p.329–349;
[13]**Hooke R., Jeeves T.A. (1961)**, ***Direct Search Solution of Numerical and Statistical Problems,*** Journal of the Association of Computing Machinery, 8, p.212-229;
[14]**Hoshino S. (1970)**, ***On Davies, Swann and Campey Minimization Process,*** The Computer Journal, 14(4), p.426-427;
[15]**Hvattum L.M., Glover F. (2009)**, ***Finding Local Optima of High-dimensional Functions Using Direct Search Methods***, European Journal of Operational Research, 195, p.31-45;
[16]**Kolda R M., Lewis R.M., Torczon V.J. (2003)**, ***Optimization by Direct Search: New Perspectives on some Classical and Modern Methods***, SIAM Review, 45, p.385-482;
[17]**Kuye A.O. (1991)**, ***Numerical Experiments with the One-Dimensional Non-Linear Simplex Search,*** Computers & Operations Research, 18(6), p.497-506;

[18]**Lewis R.M., Torczon V., Trosset M.W. (2000)**, ***Direct Search Methods: then and now,*** Journal of Computational and Applied Mathematics, 124, p.191-207;

[19]**Mladenovic N., Drazic M., Kovacevic-Vujcic V., Cangalovic M. (2008)**, ***General Variable Neighborhood Search for the Continuous Optimization***, European Journal of Operational Research, 191, p.753–770;

[20]**Nelder J.A., Mead R. (1965)**, ***A Simplex Method for Function Minimization***, The Computer Journal, 7, p.308-313;

[21]**Palmer J.R. (1969)**, ***An Improved Procedure for Orthogonalising the Search Vectors in Rosenbrock's and Swann's Direct Search Optimisation Methods***, The Computer Journal, 12, p.69-71;

[22]**Powell M.J.D. (1964)**, ***An Efficient Method for Finding the Minimum of a Function of Several Variables without Calculating Derivatives,*** The Computer Journal, 7, p.155-162;

[23]**Price C.J., Coope I.D., Byatt D. (2002)**, ***A Convergent Variant of the Nelder–Mead Algorithm,*** Journal of Optimization Theory and Applications, 113, p.5–19;

[24]**Rosenbrock H.H. (1960)**, ***An Automatic Method for Finding the Greatest or Least Value of a Function,*** The Computer Journal, 3, p.175-184;

[25]**Sacco W.F., Filho H.A., Henderson N., de Oliveira C.R.E. (2008)**, ***A Metropolis Algorithm Combined with Nelder-Mead Simplex Applied to Nuclear Core Design***, Annals of Nuclear Energy, 35, p.861-867;

[26]**Solis F.J., Wets J.B. (1981)**, ***Minimization by Random Search Techniques***, Mathematical Operations Research, 6, p.19-30;

[27]**Spall J.C. (1998)**, ***Implementation of the Simultaneous Perturbation Algorithm for Stochastic Optimization***, IEEE Transactions on Aerospace and Electronic Systems, 34, p.817-828;

[28]**Spendley W., Hext G.R., Himsworth F.R. (1962)**, ***Sequential Application of Simplex Designs in Optimisation and Evolutionary Operation,*** Technometrics, 4, p.441-461;

[29]**Swann W.H. (1964)**, ***Report on the Development of a New Direct Search Method of Optimization***, Central Instrument Laboratory Research Note 64/3., Imperial Chemical Industries Ltd.;

[30]**Torczon V. (1989)**, ***Multi-Directional Search: A Direct Search Algorithm for Parallel Machines***, PhD thesis, Rice University;

[31]**Zangwill W.I. (1967)**, ***Minimizing a Function without Calculating Derivatives,*** The Computer Journal, 10, p.293-296;

[32]**Zhao Q.H., Urosevic D., Mladenovic N., Hansen P. (2009)**, ***A Restarted and Modified Simplex Search for Unconstrained Optimization,*** Computers & Operations Research, 36, p.3263-3271.

APPENDIX

Test Problem 1 − Rosenbrock function

$$\min_{X\in\mathbb{R}^n}\sum_{i=1}^{n-1}\left[100\cdot\left(x_{i+1}-x_i^2\right)^2+(1-x_i)^2\right]$$

$$-10\le x_i\le 10, i=\overline{1..n}$$

Global Minimum

$$X^*=(1\quad 1\quad \cdots\quad 1),\ \ f^*(X)=0$$

$$n=2,4,8,16,32,64,128$$

Test Problem 2: Zakharov function

$$\min_{X\in\mathbb{R}^n}\sum_{i=1}^{n}x_i^2+\left(\sum_{i=1}^{n}0.5\cdot i\cdot x_i\right)^2+\left(\sum_{i=1}^{n}0.5\cdot i\cdot x_i\right)^4$$

$$-10\le x_i\le 10, i=\overline{1..n}$$

Global and Local Minimum

$$X^*=(0\quad 0\quad \cdots\quad 0),\ \ f^*(X)=0$$

$$n=2,4,8,16,32,64,128$$

Test Problem 3 − Matyas function

$$\min_{X\in\mathbb{R}^n}\sum_{i=1}^{n-1}0.26\cdot\left(x_i^2+x_{i+1}^2\right)-0.48\cdot x_i x_{i+1}$$

$$-10\le x_i\le 10, i=\overline{1..n}$$

Global and Local Minimum

$$X^*=(0\quad 0\quad \cdots\quad 0),\ \ f^*(X)=0$$

$$n=2,4,8,16,32,64,128,256,512$$

Test Problem 4 – Sphere function

$$\min_{X\in\mathbb{R}^n}\sum_{i=1}^{n} x_i^2$$

$$-5.12 \le x_i \le 5.12, i = \overline{1..n}$$

Global and Local Minimum

$$X^* = (0 \quad 0 \quad \cdots \quad 0),\ f^*(X) = 0$$

$$n = 2, 4, 8, 16, 32, 64, 128, 256, 512$$

Test Problem 5 – Sum Squares function

$$\min_{X\in\mathbb{R}^n}\sum_{i=1}^{n} i\cdot x_i^2$$

$$-10 \le x_i \le 10, i = \overline{1..n}$$

Global and Local Minimum

$$X^* = (0 \quad 0 \quad \cdots \quad 0),\ f^*(X) = 0$$

$$n = 2, 4, 8, 16, 32, 64, 128, 256, 512$$

Test Problem 6 – Trid function

$$\min_{X\in\mathbb{R}^n}\sum_{i=1}^{n}(x_i - 1)^2 - \sum_{i=2}^{n} x_i x_{i-1}$$

$$-n^2 \le x_i \le n^2, i = \overline{1..n}$$

Shifted Global and Local Minimum

$$f^*(X) = 0$$

$$n = 2, 4, 8, 16, 32, 64$$

Test Problem 7 – Booth function

$$\min_{X\in\mathbb{R}^n}\sum_{i=1}^{\frac{n}{2}}\left[(x_{2i-1} + 2x_{2i} - 7)^2 + (2x_{2i-1} + x_{2i} - 5)^2\right]$$

$$-10 \le x_i \le 10, i = \overline{1..n}$$

Global Minimum

$$X^* = (1 \quad 3 \quad \cdots \quad 1 \quad 3),\ f^*(X) = 0$$

$$n = 2, 4, 8, 16, 32, 64, 128, 256, 512$$

Test Problem 8 — Branin function

$$\min_{X\in\mathbb{R}^n} \sum_{i=1}^{\frac{n}{2}} \left[\left(x_{2i} - \left(\frac{5}{4\pi^2}\right) x_{2i-1}^2 + \left(\frac{5}{\pi}\right) x_{2i-1} - 6 \right)^2 + 10\left(1 - \frac{1}{8\pi}\right) \cos x_{2i-1} + 10 \right] - \frac{0.397887357729738 \cdot n}{2}$$

$$-5 \le x_i \le 10, i = \overline{1..n}$$

Multiple Global Minimum

$$f^*(X) = 0$$

$$n = 2, 4, 8, 16, 32, 64, 128, 256, 512$$