DOI: 10.1515/awutm -2016-0012
DE GRUYTER OPEN

Analele Universităţii de Vest,
Timişoara
Seria Matematică – Informatică
LIV, 2, (2016), 13– 35

Gradual and Cumulative Improvements to the Classical Differential Evolution Scheme through Experiments

George Anescu

Abstract. The paper presents the experimental results of some tests conducted with the purpose to gradually and cumulatively improve the classical *DE* scheme in both efficiency and success rate. The modifications consisted in the randomization of the scaling factor (a simple jitter scheme), a more efficient Random Greedy Selection scheme, an adaptive scheme for the crossover probability and a resetting mechanism for the agents. After each modification step, experiments have been conducted on a set of 11 scalable, multimodal, continuous optimization functions in order to analyze the improvements and decide the new improvement direction. Finally, only the initial classical scheme and the constructed Fast Self-Adaptive DE (*FSA-DE*) variant were compared with the purpose of testing their performance degradation with the increase of the search space dimension. The experimental results demonstrated the superiority of the proposed *FSA-DE* variant.

AMS Subject Classification (2000). 68T20, 68W10, 90C26, 90C56, 90C59
Keywords. Continuous Global Optimization Problem (CGOP), Derivative-free Optimization, Differential Evolution (DE), DE/rand/1/bin, Self-Adaptive DE variants, Fast Self-Adaptive DE (FSA-DE)

1 Introduction

Since its original introduction by K. Price and R. Storn in 1995 (publicly available in 1997, see [1]), *DE* has drawn the attention of many researchers all over the world resulting in numerous variants with improved performance and a significant number of engineering applications especially in optimization problems where the gradient is difficult or even impossible to derive. *DE* proved to be an advantageous method for optimizing real-valued multimodal objective functions due to its innate properties of good convergence, simplicity, ease of use, suitability for parallelization and small number of control parameters. Even from the beginning it was obvious that the performance of *DE* was greatly sensitive to the choice of the control parameters scaling factor (F), crossover probability (C_r) and population size (N), and the selection of good parameters was a challenging issue. Initial efforts have been put into finding guidelines for choosing proper *DE* control parameters for different kinds of optimization problems, see for example the pioneering work of R. Storn et al. [2], [3], J. Liu and J. Lampinen [4], or in the attempt to mathematically derive good *DE* parameters, see the pioneering work of D. Zaharie [5]. Another research direction has been devoted to eliminating the need for manual parameter tuning altogether, by automatically perturbing or adapting the control parameters during optimization based on feedback from the search, see the pioneering work of J. Liu and J. Lampinen [6], A.K. Qin et al. [7] [8], and J. Brest et al. [9]. Recently even more research has been done in the field for more complex optimizers with adaptive control parameters, see the general survey articles of Swagatam Das et al ([10], [11]), or a previous survey of F. Neri and V. Tirronen ([12]), where special sections are dedicated to self-adaptive variants of *DE*.

The purpose of the present paper is to build a Fast Self-Adaptive variant of *DE* (*FSA-DE*) by starting from the classical scheme *DE/rand/1/bin* with fixed parameters and testing various improvement ideas in a gradual and cumulative manner through testing experiments.

The rest of this paper is organized as follows: Section 2 presents the Continuous Global Optimization Problem (*CGOP*); Section 3 presents the Deb's Rules for Constraints Handling as they were adapted for the *FSA-DE* method; Section 4 presents the principles of the original *DE/rand/1/bin* scheme; Section 5 presents the testing methodology adopted for comparing the effects of different modifications implemented in the *FSA-DE* method and the set of test optimization problems used in the experiments; Section 6 presents the first improvement implemented in the *FSA-DE* method,

the randomization of parameter F, and the obtained experimental results; Section 7 presents the second improvement implemented in the *FSA-DE* method, the Random Greedy Selection (*RGS*), and the obtained experimental results; Section 8 presents the third improvement implemented in the *FSA-DE* method, the randomization of parameter C_r, and the obtained experimental results; Section 9 presents the fourth improvement implemented in the *FSA-DE* method, the improved adaptive randomization of parameter C_r, and the obtained experimental results; Section 10 presents the fifth and final improvement implemented in the *FSA-DE* method, the resetting mechanism, and the obtained experimental results; Section 11 presents the comparison between the original *DE/rand/1/bin* scheme and the *FSA-DE* method when the dimension of the search space increases; and finally Section 12 summarizes and draws some conclusions.

2 Continuous Global Optimization Problem (*CGOP*)

The Continuous Global Optimization Problem (*CGOP*) is generally formulated as ([13]):

$$\begin{aligned} &\text{minimize} && f(\mathbf{x}) \\ &\text{subject to} && \mathbf{x} \in D \end{aligned} \tag{2.1}$$

with

$$\begin{aligned} D = \{\mathbf{x} : \mathbf{l} \leq \mathbf{x} \leq \mathbf{u};\ &and\ \ g_i(\mathbf{x}) \leq 0,\ \ i = 1, \ldots, G; \\ &and\ \ h_j(\mathbf{x}) = 0,\ \ j = 1, \ldots, H\} \end{aligned} \tag{2.2}$$

where $\mathbf{x} \in \mathbb{R}^n$ is a real n-dimensional vector of decision variables, $f : \mathbb{R}^n \to \mathbb{R}$ is the continuous objective function, $D \subset \mathbb{R}^n$ is the non-empty set of feasible decisions (a proper subset of $\mathbb{R}^n$), $\mathbf{l}$ and $\mathbf{u}$ are explicit, finite (component-wise) lower and upper bounds on $\mathbf{x}$, $g_i : \mathbb{R}^n \to \mathbb{R},\ i = 1, \ldots, G$ is a finite collection of continuous inequality constraint functions, and $h_j : \mathbb{R}^n \to \mathbb{R},\ j = 1, \ldots, H$ is a finite collection of continuous equality constraint functions. No other additional supposition is made on the *CGOP* problem and it is assumed that no additional knowledge about the collections of real continuous functions can be obtained, in this way treating the *CGOP* problem as a black box, i.e. for any point $\mathbf{x}$ in the boxed domain $\{\mathbf{x} : \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}\}$ it is assumed the ability to calculate the values of the functions $f(\mathbf{x})$, $g_i(\mathbf{x}),\ i = 1, \ldots, G$, $h_j(\mathbf{x}),\ j = 1, \ldots, H$, but nothing more.

3 Deb's Rules for Constraints Handling

The Deb's rules ([14]) offer a methodology to efficiently handle the constraints in constrained optimization problems. The presentation in this section is adapted from [15] with the notations from equations (2.1) and (2.2). The inequality constraints that satisfy $g_i(\mathbf{x}) = 0,\ i = 1, \ldots, G$ at the global optimum solution are called active constraints. All equality constraints are active constraints. The equality constraints can be transformed into the inequality form and can be combined with other inequality constraints as the auxiliary functions $\tilde{g}_i(\mathbf{x})$:

$$\tilde{g}_i(\mathbf{x}) = \begin{cases} \max[0, g_i(\mathbf{x})], & i = 1, \ldots, G \\ \max[0, |h_{i-G}(\mathbf{x})| - \delta], & i = G+1, \ldots, G+H. \end{cases} \tag{3.1}$$

where δ is a tolerance parameter for the equality constraints. Therefore, the objective is to minimize the objective function $f(\mathbf{x})$ such that the obtained optimal solution satisfies all the inequality constraints $\tilde{g}_i(\mathbf{x}) \leq 0$ as active constraints. The overall constraint violation for an infeasible solution is a weighted mean of all the constraints expressed as:

$$v(\mathbf{x}) = \frac{\sum_{i=1}^{G+H} w_i \tilde{g}_i(\mathbf{x})}{\sum_{i=1}^{G+H} w_i} \tag{3.2}$$

where $w_i = 1/\tilde{g}_{max,i}$ is a weight parameter and $\tilde{g}_{max,i}$ is the maximum violation of constraint $\tilde{g}_i(\mathbf{x})$ obtained so far (up to the current iteration). $\tilde{g}_{max,i}$ varies during the optimization process in order to balance the contribution of every constraint in the problem irrespective of their differing numerical ranges. There are a number of constraint handling techniques based on constraint violation, the one used here being the Superiority of Feasible Solutions (SF) technique. In SF the Deb's rules ([14]) are applied when comparing two solutions $\mathbf{x}_{i_1}$ and $\mathbf{x}_{i_2}$. According to Deb's rules $\mathbf{x}_{i_1}$ is regarded superior to $\mathbf{x}_{i_2}$ when:

- $\mathbf{x}_{i_1}$ is feasible and $\mathbf{x}_{i_2}$ is not feasible.
- $\mathbf{x}_{i_1}$ and $\mathbf{x}_{i_2}$ are both feasible and $\mathbf{x}_{i_1}$ has a smaller objective value than $\mathbf{x}_{i_2}$.
- $\mathbf{x}_{i_1}$ and $\mathbf{x}_{i_2}$ are both infeasible, but $\mathbf{x}_{i_1}$ has a smaller overall constraint violation than $\mathbf{x}_{i_2}$.

Therefore, in SF the feasible solutions are always considered better than the infeasible ones. Two infeasible solutions are compared based on their overall constraint violations only, while two feasible solutions are compared based on their objective function values only. The comparison of infeasible solutions based on the overall constraint violation aims to push the infeasible solutions toward the feasible regions, while the comparison of two feasible solutions based on the objective function value improves the overall solution.

In order to be able to correctly compare the infeasible solutions which are near feasible regions the following modification to the constraint violation was proposed:

$$v(\mathbf{x}) = \frac{\sum_{i=1}^{G+H} w_i \tilde{g}_i(\mathbf{x})}{\sum_{i=1}^{G+H} w_i} + G_{ns} + H_{ns} \tag{3.3}$$

where G_{ns} ($G_{ns} \leq G$) is the number of not satisfied inequality constraints, and H_{ns} ($H_{ns} \leq H$) is the number of not satisfied equality constraints.

Another important proposed improvement for the handling of the equality constraints is to make the δ tolerance parameter dependent on the current iteration index k:

$$\delta = k(\delta_2 - \delta_1)/iter_{max} + \delta_1 \tag{3.4}$$

where δ_1 is the initial tolerance parameter (at $k = 0$), δ_2 is the final tolerance parameter (at $k = iter_{max}$) with $\delta_1 \gg \delta_2$, and $iter_{max}$ is the maximum iteration index.

4 Differential Evolution scheme *DE/rand/1/bin*

This study is starting from the classical Differential Evolution scheme, *DE/rand/1/bin* (sometimes called scheme *DE1*), proposed by K. Price, R. Storn in 1997 ([1]). In the scheme's name *DE* comes from Differential Evolution, *rand* comes from random selection, 1 is the number of vector differences used in the mutation equation and *bin* comes from binomial crossover (or recombination). Like other Evolutionary Strategy population-based methods, DE generates new points that are perturbations of existing points, but unlike other Evolutionary Strategy methods, DE perturbs population vectors with the scaled difference of two randomly selected

population vectors to produce the trial vectors. The simplified pseudo-code of the implemented DE algorithm is presented below:

Step 1: Initialization;
while (true)
    Step 2: Check termination conditions;
    Step 3: Mutation;
    Step 4: Crossover;
    Step 5: Evaluation
    Step 6: Selection;
end while

The method's parameters are: N the number of evolving agents (or population size), F the scaling factor, C_r the crossover probability, ϵ the required precision and $iter_{max}$ the maximum number of iterations. The N agents are represented as vector positions, $\mathbf{x}_i,\ \ i = 1, \ldots, N$, in the limiting box (hyper-rectangle) defined by the lower limits $l_j,\ \ j = 1, \ldots, n$ and higher limits $h_j,\ \ j = 1, \ldots, n$ $(l_j < h_j)$. The agents' vector positions are considered possible solutions to the optimization problem. At initialization the agents take random values in the limiting box:

$$x_{i,j}(0) = l_j + rnd_{i,j}(u_j - l_j),\ \ i = 1, \ldots, N,\ \ j = 1, \ldots, n \tag{4.1}$$

where $rnd_{i,j}$ are pseudo-random numbers uniformly generated in the interval $[0, 1)$. The iteration index is initialized to $k = 0$. The objective function $f(\mathbf{x})$ is evaluated in the current agent positions $\mathbf{x}_i(0),\ \ i = 1, \ldots, N$: $f_i(0) = f(\mathbf{x}_i(0)),\ \ i = 1, \ldots, N$.

A first termination condition is defined when the current iteration index k attains the maximum number of iterations $iter_{max}$. A second termination condition is defined when the diameter of the current population of agents becomes less than the required precision ϵ:

$$d(k) = \left(\sum_{j=1}^{n} (d_j(k))^2 \right)^{\frac{1}{2}} < \epsilon \tag{4.2}$$

where the overall population diameter $d(k)$ is calculated according to the Euclidian distance, and the diameters on each dimension are calculated as the maximum absolute difference between two position values on that dimension over all the agents in the population:

$$d_j(k) = \max_{1 \le i_1, i_2 \le N, i_1 \ne i_2} \{|x_{i_1,j}(k) - x_{i_2,j}(k)|\}, \quad j = 1, \ldots, n \tag{4.3}$$

A still further termination condition is defined when a flat region is detected. It can appear when the objective function $f(\mathbf{x})$ depends only on a subset of its parameters, and it can be easily checked as:

$$f_{max}(k) - f_{min}(k) < \epsilon_f \tag{4.4}$$

with

$$f_{max}(k) = \max_{1 \le i_1 \le N} \{f_{i_1}(k)\} \tag{4.5}$$

$$f_{min}(k) = \min_{1 \le i_2 \le N} \{f_{i_2}(k)\} \tag{4.6}$$

where $f_i(k) = f(\mathbf{x}_i(k))$, $i = 1, \ldots, N$ and ϵ_f is a very small value. If any of the termination conditions is satisfied the iterative process is stopped (the loop is broken) and the agent which gives $f_{min}(k)$ positioned in $\mathbf{x}_{min}(k)$ is taken as the solution to the global optimization problem. Otherwise the iteration index is incremented to $k+1$ and the computation continues to the next iteration.

The differential mutation adds a scaled, randomly sampled, vector difference to a third randomly sampled vector to create a mutant vector:

$$\mathbf{v}_i(k+1) = \mathbf{x}_{r_3}(k) + F(\mathbf{x}_{r_1}(k) - \mathbf{x}_{r_2}(k)) \;\; i = 1, \ldots, N \tag{4.7}$$

The scale factor, $F \in (0, 1+)$, is a positive real number that controls the rate at which the population evolves. While there is no upper limit on F, effective values are seldom greater than 1. The base vector index, r_3, can be determined in a variety of ways, but for now it is assumed to be a randomly chosen vector with an index that is different from the target vector's index, i. Except for being distinct from each other and from both the base and target vector indexes ($i \ne r_1 \ne r_2 \ne r_3$), the difference vector indexes, r_1 and r_2, are also randomly selected once per mutation.

To complement the differential mutation search strategy, DE also employs the binomial (or uniform) crossover. Sometimes referred to as discrete recombination, the binomial crossover builds trial vectors out of parameter values that have been copied from two different vectors. In particular, DE crosses each vector with the mutant vector:

$$u_{i,j}(k+1) = \begin{cases} v_{i,j}(k+1) & \text{if } rnd_{i,j} < C_r \text{ or } j = j_{rand} \\ x_{i,j}(k) & \text{otherwise} \end{cases} \tag{4.8}$$

$$i = 1, \ldots, N, \;\; j = 1, \ldots, n$$

The crossover probability, $C_r \in [0, 1)$, is a user-defined value that controls the fraction of parameter values that are copied from the mutant. In addition, the trial vector component with randomly chosen index, j_{rand}, is taken from the mutant to ensure that the trial vector does not duplicate $\mathbf{x}_i(k)$. Due to this additional demand, C_r only approximates the true probability that a trial parameter will be inherited from the mutant. The objective function $f(\mathbf{x})$ is evaluated in all the new proposed trial vectors $\mathbf{u}_i(k+1)$, $i = 1, \ldots, N$. If the trial vector, $\mathbf{u}_i(k+1)$, has an objective function value less than or equal to that of its target vector, $\mathbf{x}_i(k)$, it replaces its target vector in the next iteration; otherwise, the target retains its place in the population for at least one more iteration:

$$\mathbf{x}_i(k+1) = \begin{cases} \mathbf{u}_i(k+1) & \text{if } f(\mathbf{u}_i(k+1)) \leq f(\mathbf{x}_i(k)) \\ \mathbf{x}_i(k) & \text{otherwise} \end{cases} \quad i = 1, \ldots, N \tag{4.9}$$

By comparing each trial vector with the target vector from which it inherits components, *DE* more tightly integrates recombination and selection than do other Evolutionary Algorithms.

5 Testing Methodology

The purpose of the testing experiments was to prove that some gradual modifications applied to the classical scheme *DE/rand/1/bin* are producing cumulative improvements in the performance of the modified algorithm. As an initial reference for comparison, the classical scheme *DE/rand/1/bin* with fixed balanced values for the control parameters, $F = 0.5$ and $C_r = 0.5$, was chosen, and as execution mode the asynchronous mode was chosen.

The asynchronous mode is considered advantageous over the synchronous mode because it permits the improved solutions to contribute to the evolution immediately and in this way can speed up the convergence of the algorithm (see [11]). Other advantages of the asynchronous mode are the reduced memory requirements for the population (in opposition to the synchronous

mode, it does not need a buffer with the previous population) and its inherent suitability for parallelization.

After each modification the results of the current modification were compared side by side with the results of the previous algorithmic variant (without the current modification).

In order to conduct the tests an appropriate testing methodology was devised (see also [16], [17]). When the quality of an optimization method is estimated, two often conflicting characteristics are of interest: a small number of function evaluations, NFE, and a high success rate, SR. For test functions with known solutions the success can be simply defined as the achievement of an absolute or relative precision tolerance to the known solutions. By fixing the tolerance and choosing $iter_{max}$ high enough so that it is never attained before the tolerance is attained, it becomes easy to measure the SR and average NFE to success ($\mu(NFE)$). When SR is not expressed as a percentage, it can also be interpreted as the probability of success for a single run of the method for the considered test function in fixed run conditions. There are other testing methodologies frequently applied in practice, like for example based on fixing NFE and reporting the best, the worst and the median results obtained after the fixed NFE is exhausted, but in our opinion such methodologies are not recognizing the importance of success rate and are concealing it from reporting. A very efficient method (with a fast convergence), but with a low success rate, cannot be considered better than a less efficient method, but with a high success rate, because the former may need many repeated runs in order to obtain the correct result, while the later may get the correct result in less runs, which can entail a larger overall NFE (obtaining by summation) for the former compared to the later.

A test bed of 11 known scalable multimodal optimization functions (10 unconstrained and 1 constrained, see [18], [19], [20], [21], [22]) were used for the tests run on the initial *DE* scheme and the gradually improved algorithm:

- *Rastrigin's Function* - highly multimodal with the locations of the minima regularly distributed, global minimum value of 0 at $(0, 0, \ldots, 0)$

$$\begin{aligned} f_1(\mathbf{x}) &= 10n + \sum_{j=1}^{n} [x_j^2 - 10\cos(2\pi x_j)], \\ &-5.12 \leq x_j \leq 5.12, \;\; j = 1, \ldots, n \end{aligned} \tag{5.1}$$

- *Alpine 1 Function* - highly multimodal, global minimum value of 0 at $(0, 0, \ldots, 0)$

$$f_2(\mathbf{x}) = \sum_{j=1}^{n}(|x_j \sin(x_j)| + 0.1|x_j|), \\ -10 \leq x_j \leq 10, \;\; j = 1, \ldots, n \tag{5.2}$$

- *Alpine 2 Function* - highly multimodal, global maximum value of 2.808^n at $(7.917, 7.917, \ldots, 7.917)$

$$f_3(\mathbf{x}) = \prod_{j=1}^{n} \sqrt{x_j} \sin(x_j), \\ 0 \leq x_j \leq 10, \;\; j = 1, \ldots, n \tag{5.3}$$

- *Griewangk's Function* - many widespread local minima regularly distributed with the global minimum of 0 at $(0, 0, \ldots, 0)$

$$f_4(\mathbf{x}) = \frac{1}{4000} \sum_{j=1}^{n} x_j^2 - \prod_{j=1}^{n} \cos\left(\frac{x_j}{\sqrt{j}}\right) + 1, \\ -100 \leq x_j \leq 100, \;\; j = 1, \ldots, n \tag{5.4}$$

- *Schwefel's Function* - many widespread local minima distributed at distance from the origin with the global minimum of -418.9829 at $(420.9687, 420.9687, \ldots, 420.9687)$

$$f_5(\mathbf{x}) = -\frac{1}{n} \sum_{j=1}^{n} x_j \sin\left(\sqrt{|x_j|}\right), \\ -500 \leq x_j \leq 500, \;\; j = 1, \ldots, n \tag{5.5}$$

- *Paviani's Function* - many local minima with the global minimum of -45.77847 at $(9.351, 9.351, \ldots, 9.351)$ for $n = 10$, -9549.89061 at $(9.9658, 9.9658, \ldots, 9.9658)$ for $n = 20$, and respectively -99786.45525 at $(9.9993, 9.9993, \ldots, 9.9993)$ for $n = 30$:

$$f_6(\mathbf{x}) = \sum_{j=1}^{n-1} \left[\log(x_j - 2)^2 + \log(10 - x_j)^2\right] - \left(\prod_{j=1}^{n-1} x_j\right)^{0.2}, \quad 2.0001 \leq x_j \leq 9.9999, \;\; j = 1, \ldots, n \tag{5.6}$$

- *Expanded Schaffer's Function* - many local minima with the global minimum of 0 at $(0, 0, \ldots, 0)$

$$f_7(\mathbf{x}) = g(x_1, x_2) + g(x_2, x_3) + \ldots + g(x_n, x_1), \quad -10 \leq x_j \leq 10, \;\; j = 1, \ldots, n \tag{5.7}$$

where

$$g(x, y) = 0.5 + \frac{\sin^2(\sqrt{x^2 + y^2}) - 0.5}{1 + 0.001(x^2 + y^2)^2} \tag{5.8}$$

- *Michaelwitz's Function* - highly multimodal with global minimum of: -0.966015 for $n = 10$, -0.9818507 for $n = 20$, and respectively -0.9876481 for $n = 30$:

$$f_8(\mathbf{x}) = -\frac{1}{n} \sum_{j=1}^{n} \sin(x_j) \sin^{2m}\left(\frac{j x_j^2}{\pi}\right), \quad m = 10, \;\; 0 \leq x_j \leq \pi, \;\; j = 1, \ldots, n \tag{5.9}$$

- *Ackley's Function* - highly multimodal with global minimum of 0 at $(0, 0, \ldots, 0)$:

$$f_9(\mathbf{x}) = 20 + e - 20e^{-0.2\left(\frac{1}{n}\sum_{j=1}^{n} x_j^2\right)^{1/2}} - e^{\frac{1}{n}\sum_{j=1}^{n} cos(2\pi x_j)}, \quad -30 \leq x_j \leq 30, \;\; j = 1, \ldots, n \tag{5.10}$$

- *Non-Linear Function* - highly multimodal, many global minima of 0:

$$f_{10}(\mathbf{x}) = n - 1 + \sum_{j=1}^{n-1} \cos\left(\frac{|x_{j+1} - x_j|}{|x_j + x_{j+1}| + 10^{-10}}\right), \\ -10 \leq x_j \leq 10, \;\; j = 1, \ldots, n \tag{5.11}$$

- *Keane's Bump Function* - highly multimodal open constrained problem, best known global minima for different search space dimensions were used during testing (-0.747310362 for $n = 10$, -0.803619104 for $n = 20$, and respectively -0.821878040697 for $n = 30$):

$$f_{11}(\mathbf{x}) = -\left|\left\{\sum_{j=1}^{n}\cos(x_j)^4 - 2\prod_{j=1}^{n}\cos(x_j)^2\right\} \Big/ \left(\sum_{j=1}^{n} j x_j^2\right)^{0.5}\right|, \\ g_1(\mathbf{x}) = 0.75 - \prod_{j=1}^{n} x_j \leq 0.0, \\ g_2(\mathbf{x}) = \sum_{j=1}^{n} x_j - 7.5n \leq 0.0, \\ 0.0 \leq x_j \leq 10.0, \;\; j = 1, \ldots, n \tag{5.12}$$

  Note that the global minimum -0.821878040697 for $n = 30$ was found during testing and it is an improvement over -0.818056222 reported in [22].

All the test functions with global optima in origin (f_1, f_2, f_4, f_7 and f_9) were shifted, considering that the origin is favored by *DE* type methods and this peculiarity can interfere in the results. The origins for the dimensions were shifted to the corresponding components of the position **x0** with:

$$x0_j = l_j + \frac{j(u_j - l_j)}{n+1}, \;\; j = 1, \ldots, n. \tag{5.13}$$

The ultimate purpose of the improvements tested in this study was to eliminate the dependence of the algorithm on fixed control parameters F and C_r, but in the process care has to be taken in order to not introduce new parameters that need tuning from the part of the practitioner, and in this

way only deferring the problem of parameter elimination. Related to the third control parameter, the population size N, the recommendation from K. Price and R. Storn was taken into consideration (see [1]) that N should be reasonably chosen between $5 \times n$ and $10 \times n$ (n being the dimension of the search space), but for the constrained test function even larger N than recommended was considered. The tests on the gradual improvements were conducted only for $n = 10$ and for the first 10 unconstrained test functions, but for the final tests, where the purpose was to investigate how the final *FSA-DE* algorithm compares to the initial *DE/rand/1/bin* scheme when n increases ($n = 20$ and $n = 30$), the 11-th constrained test function was also introduced in the testbed.

6 First Improvement, Randomizing the Scale Factor

In order to eliminate the F control parameter from equation (4.7), as the first modification, was tested the randomization in the real interval $[0, 1)$ of the F control parameter for all the components of the vector difference. This type of modification is a particular case of a more general modification known in the *DE* literature as Jitter scheme (see [3], [23]), where F is pseudo-randomly modified for all vector components around a midpoint and within a defined range, for example, in our case the midpoint is 0.5 and the range is 0.5, but in order to keep the explanation simple it can be better interpreted that the vector difference in equation (4.7) is between a pseudo-randomly point selected inside the hyper-rectangle defined by the two agent positions $\mathbf{x}_{r_1}$ and $\mathbf{x}_{r_2}$ (the agent positions being the end points of a main diagonal of the hyper-rectangle), and position $\mathbf{x}_{r_2}$:

$$\mathbf{v}_i(k+1) = \mathbf{x}_{r_3}(k) + \mathbf{r}_i \otimes (\mathbf{x}_{r_1}(k) - \mathbf{x}_{r_2}(k)) \;\; i = 1, \ldots, N \tag{6.1}$$

where $\mathbf{r}_i$ is a vector of pseudo-random real numbers uniformly distributed in the $[0, 1)$ real interval and the symbol $\otimes$ has the meaning of component by component vector product.

Table 1 presents the comparative testing results between the initial scheme *DE/rand/1/bin* with fixed parameters, $F = 0.5$ and $C_r = 0.5$, and *FSA-DE* with the first modification, for $n = 10$. From the success rate perspective can be observed that *FSA-DE* performed better for f_{10}, while for the other test functions both methods produced excellent results. From the efficiency perspective the results were mixed, *FSA-DE* surpassed the initial scheme for f_1, f_3, f_5, f_7, f_8 and f_{10}, but not for the other test functions,

Table 1: *FSA-DE* first improvement versus *DE/rand/1/bin* with fixed parameters $F = 0.5$ and $C_r = 0.5$, $n = 10$, $runs = 100$, $tolerance = 0.1\%$, $N = 100$

Fn	$SR\%$ DE/rand/1/bin	$\mu(NFE)$ DE/rand/1/bin	$SR\%$ FSA-DE first	$\mu(NFE)$ FSA-DE first
f_1	100%	78339	100%	51548
f_2	100%	15527	100%	15868
f_3	100%	22553	100%	18448
f_4	100%	81007	100%	83592
f_5	100%	13652	100%	12811
f_6	100%	6553	100%	6780
f_7	100%	419626	100%	239754
f_8	98%	42323	98%	35352
f_9	100%	19743	100%	21046
f_{10}	65%	599369	87%	443683

although when *FSA-DE* was surpassed the differences were relatively small. Overall, the improvements may seem small, but the important achievement is that the parameter F was eliminated almost without any degradation in performance. Also, as n increases the modification can provide a better exploration of the search space, considering that for fixed F the number of choices for new positions is limited, since the number of arrangements (r_1, r_2, r_3) is limited and each arrangement provides only one new possible position, while with the randomization scheme can be explored continuous portions of the search space.

7 Second Improvement, Random Greedy Selection

The index r_3 in equation (6.1) was previously selected as a pseudo-random index different from the target agent index i. In order to increase the efficiency, at this step a different selection method, Random Greedy Selection (*RGS*) (see [16]), was considered. In *RGA* the index r_3 is pseudo-randomly selected (according to the uniform discrete distribution) from the set l_i of indexes of agents with better fitness than the target agent with index i:

$$l_i = \{r_3 : 1 \leq r_3 \leq N;\ \ r_3 \neq i;\ \ and\ \ f(\mathbf{x}_{r_3}) < f(\mathbf{x}_i)\} \tag{7.1}$$

Table 2 presents the comparative testing results, obtained for $n = 10$, between the first scheme of *FSA-DE* and the second scheme of *FSA-DE*. From the success rate perspective the results were mixed, it can be observed that the second scheme of *FSA-DE* was better than the first scheme of *FSA-DE* for f_{10}, but it lost in performance for f_4 and f_8, the loss for f_8 being more serious. For the other test functions both methods produced the maximum

Table 2: *FSA-DE* second improvement versus *FSA-DE* first improvement, $n = 10$, $runs = 100$, $tolerance = 0.1\%$, $N = 100$

Fn	$SR\%$ FSA-DE first	$\mu(NFE)$ FSA-DE first	$SR\%$ FSA-DE second	$\mu(NFE)$ FSA-DE second
f_1	100%	51548	100%	33803
f_2	100%	15868	100%	11096
f_3	100%	18448	100%	12643
f_4	100%	83592	98%	51814
f_5	100%	12811	100%	9118
f_6	100%	6780	100%	5233
f_7	100%	239754	100%	123739
f_8	98%	35352	85%	24178
f_9	100%	21046	100%	14299
f_{10}	87%	443683	99%	192980

success rate percentage. From the efficiency perspective the results were excellent with the efficiency increasing for all test functions, notably for f_7 and f_{10}.

8 Third Improvement, Randomizing the Crossover Probability

The main problems identified in Table 2 consist in a still low efficiency for test functions f_7 and f_{10} and a relatively low success rate for test function f_8 (when compared with the results obtained for the other test functions). In order to fix these problems, without affecting the other test functions, here was tested the randomization of the crossover probability C_r by selecting it in an uniformly pseudo-random manner from the real interval $[0, 1)$, independently for each agent.

Table 3: *FSA-DE* third improvement versus *FSA-DE* second improvement, $n = 10$, $runs = 100$, $tolerance = 0.1\%$, $N = 100$

Fn	$SR\%$ FSA-DE second	$\mu(NFE)$ FSA-DE second	$SR\%$ FSA-DE third	$\mu(NFE)$ FSA-DE third
f_1	100%	33803	100%	28596
f_2	100%	11096	100%	11904
f_3	100%	12643	100%	13580
f_4	98%	51814	99%	47395
f_5	100%	9118	100%	9963
f_6	100%	5233	100%	5512
f_7	100%	123739	100%	85812
f_8	85%	24178	93%	23886
f_9	100%	14299	100%	15309
f_{10}	99%	192980	98%	163478

Table 3 presents the comparative testing results, obtained for $n = 10$,

between the second scheme of *FSA-DE* and the third scheme of *FSA-DE*. From the success rate perspective the results were mixed, with a small increase for f_4, a small decrease for f_{10} and a more important increase for f_8. The efficiency was further improved for f_7 and f_{10} with small mixed differences for the rest of test functions: relatively small increases for f_1, f_4 and f_8, and relatively small decreases for f_2, f_3, f_5, f_6 and f_9.

9 Fourth Improvement, Adaptive Gaussian Distribution for Crossover Probability

The randomization of C_r implemented in the third *FSA-DE* step produced overall only a small improvement. In the fourth improvement the idea to use a probability distribution for sampling the crossover probability C_r was kept, but the uniform probability distribution was replaced with a gaussian probability distribution which is using the fitness improvement to guide its evolution during the optimization process. Initially, at iteration $k = 0$, was used the probability distribution $C_r \sim N(\mu(0), \sigma(0))$ with $\mu(0) = 0.5$ and $\sigma(0) = 0.25$, then it was recursively evolved at iteration k to $N(\mu(k), \sigma(k))$. At each iteration a specific crossover probability was associated to each agent by sampling the probability distribution calculated at the previous step, for example considering that the current iteration index is $k+1$, the associated crossover probabilities $cr_i(k+1)$ are sampled according to the probability distribution at the previous iteration, $C_r \sim N(\mu(k), \sigma(k))$. After evolving all the agents, a fitness improvement was associated to each agent:

$$\Delta f_i(k+1) = f(\mathbf{x}_i(k)) - f(\mathbf{x}_i(k+1)) \tag{9.1}$$

Note that for the agents which are not improving at step $k+1$, the fitness improvements are 0, but they are not affecting the computations that follow. The updated $\mu(k+1)$ and $\sigma(k+1)$ are computed according to:

$$\mu(k+1) = \frac{\sum_{i=1}^{N} cr_i(k+1)\Delta f_i(k+1)}{\sum_{i=1}^{N} \Delta f_i(k+1)} \tag{9.2}$$

$$\sigma(k+1) = \left(\frac{\sum_{i=1}^{N} \Delta f_i(k+1)(cr_i(k+1) - \mu(k+1))^2}{\sum_{i=1}^{N} \Delta f_i(k+1)} \right)^{\frac{1}{2}} \tag{9.3}$$

where each agent's crossover probability is weighted proportionally to its fitness improvement, in this way the agents with better fitness improvement contributing more substantially to the shaping of the new probability distribution, $N(\mu(k+1), \sigma(k+1))$.

Other rules applied are that $\sigma(k+1)$ is limited to the real interval $[0.05, 0.25]$, from considerations that for $\sigma > 0.25$ the gaussian is becoming too flat and for $\sigma < 0.05$ it is becoming to narrow, and both conditions may impede the correct dynamic evolution of the probability distribution. If the number of improved agents becomes less than 5% of the population size N, then for the next step the probability distribution is switched to the uniform probability distribution $C_r \sim U(0, 1)$, from considerations that a stagnation state was attained and the data are insufficient to correctly evolve the gaussian probability distribution.

Table 4: *FSA-DE* fourth improvement versus *FSA-DE* third improvement, $n = 10$, $runs = 100$, $tolerance = 0.1\%$, $N = 100$

Fn	$SR\%$ FSA-DE third	$\mu(NFE)$ FSA-DE third	$SR\%$ FSA-DE fourth	$\mu(NFE)$ FSA-DE fourth
f_1	100%	28596	100%	18628
f_2	100%	11904	100%	11259
f_3	100%	13580	100%	13848
f_4	99%	47395	100%	33508
f_5	100%	9963	100%	9108
f_6	100%	5512	100%	5025
f_7	100%	85812	100%	89892
f_8	93%	23886	98%	16604
f_9	100%	15309	100%	14069
f_{10}	98%	163478	100%	191337

Table 4 presents the comparative testing results, obtained for $n = 10$, between the third scheme of *FSA-DE* and the fourth scheme of *FSA-DE*. From the success rate perspective the results were excellent, it produced maximum success rate percentages for all test functions with the exception of f_8, but the increase for f_8 was notable. For efficiency the results were mixed with notable increases for f_1, f_2, f_4, f_5, f_6, f_8 and f_9, a small decrease for f_3, but decreases for f_7 and f_{10}.

10 Fifth Improvement, the Resetting Mechanism

The main problem at this step was the relatively reduced efficiency of *FSA-DE* for f_7 and f_{10}. It was assumed that it is due to a stagnation phenomenon which occurs for these two test functions in the optimization process, and in order to mitigate the problem, an idea inspired from the Scout Bees Phase in Artificial Bee Colony (ABC) optimization method (see [24]) was used. A *limit* parameter was introduced, similar to the one used in *ABC*, and a stagnation count, associated to each agent, was updated at each iteration. At each iteration at most one agent was reset, the one with the highest stagnation count, but only if its stagnation count was higher than *limit*. Some recommendations are made in the original *ABC* paper ([24]) for setting the *limit* parameter, but here we preferred a more dynamic setting also tested in [16], $limit = 4 \times n$. Also, for further improvement in efficiency, the reset position of the agent is determined in a uniformly pseudo-random manner inside the current hyper-rectangle that is limiting the agents (also recommended in [16]) with lower and upper limits $\mathbf{l}(k+1)$ and respectively $\mathbf{u}(k+1)$ determined on each dimension:

$$l_j(k+1) = \min_{1 \le i \le N} \{x_{i,j}(k+1)\} \tag{10.1}$$

$$u_j(k+1) = \max_{1 \le i \le N} \{x_{i,j}(k+1)\} \tag{10.2}$$

Table 5: *FSA-DE* fifth improvement (final) versus *FSA-DE* fourth improvement, $n = 10$, $runs = 100$, $tolerance = 0.1\%$, $N = 100$

Fn	$SR\%$ FSA-DE fourth	$\mu(NFE)$ FSA-DE fourth	$SR\%$ FSA-DE fifth	$\mu(NFE)$ FSA-DE fifth
f_1	100%	18628	100%	18830
f_2	100%	11259	100%	11239
f_3	100%	13848	100%	14276
f_4	100%	33508	100%	34603
f_5	100%	9108	100%	9245
f_6	100%	5025	100%	5125
f_7	100%	89892	100%	66707
f_8	98%	16604	98%	16897
f_9	100%	14069	100%	14049
f_{10}	100%	191337	100%	80964

Table 5 presents the comparative testing results, obtained for $n = 10$, between the fourth improvement scheme of *FSA-DE* and the fifth improvement scheme of *FSA-DE*. It can be observed from the table that the success rate was not affected by the modification, but the efficiency was notably improved for the problematic functions f_7 and f_{10}, which is

a proof that the assumption related to the stagnation was correct. The other test functions were very little affected with mixed results: the efficiency somewhat increased for f_2 and f_9, but it somewhat decreased for the other test functions.

11 Increasing the Search Space Dimension

In the last phase of the testing experiments the purpose was to investigate how the performance of the novel *FSA-DE* method is affected by the increase of the search space dimension n, and to compare it with the performance of the reference scheme *DE/rand/1/bin* with fixed parameters, $F = 0.5$ and $C_r = 0.5$. The constrained test function f_{11} was also introduced in the testbed.

Table 6: *FSA-DE* versus *DE/rand/1/bin* with fixed parameters $F = 0.5$ and $C_r = 0.5$, $n = 10$, $runs = 100$, $tolerance = 0.1\%$, $N = 100$

Fn	$SR\%$ DE/rand/1/bin	$\mu(NFE)$ DE/rand/1/bin	$SR\%$ FSA-DE	$\mu(NFE)$ FSA-DE
f_1	100%	78339	100%	18830
f_2	100%	15527	100%	11239
f_3	100%	22553	100%	14276
f_4	100%	81007	100%	34603
f_5	100%	13652	100%	9245
f_6	100%	6553	100%	5125
f_7	100%	419626	100%	66707
f_8	98%	42323	98%	16897
f_9	100%	19743	100%	14049
f_{10}	65%	599369	100%	80964
f_{11}^*	100%	35656	100%	25316

Table 7: *FSA-DE* versus *DE/rand/1/bin* with fixed parameters $F = 0.5$ and $C_r = 0.5$, $n = 20$, $runs = 100$, $tolerance = 0.1\%$, $N = 100$

Fn	$SR\%$ DE/rand/1/bin	$\mu(NFE)$ DE/rand/1/bin	$SR\%$ FSA-DE	$\mu(NFE)$ FSA-DE
f_1	100%	392308	100%	39458
f_2	100%	37684	100%	25500
f_3	100%	55030	100%	33163
f_4	99%	45719	99%	27546
f_5	100%	34286	100%	18980
f_6	100%	12867	100%	9283
f_7	0%	N/A	100%	307805
f_7^*	1%	1347300	N/A	N/A
f_8	90%	1116672	98%	58203
f_9	100%	45965	100%	30869
f_{10}	12%	962341	54%	379477
f_{11}^{**}	100%	95874	100%	95189

Table 8: *FSA-DE* versus *DE/rand/1/bin* with fixed parameters $F = 0.5$ and $C_r = 0.5$, $n = 30$, $runs = 100$, $tolerance = 0.1\%$, $N = 100$

Fn	$SR\%$ DE/rand/1/bin	$\mu(NFE)$ DE/rand/1/bin	$SR\%$ FSA-DE	$\mu(NFE)$ FSA-DE
f_1	5%	1389280	100%	62816
f_2	100%	67173	100%	40827
f_3	100%	122259	100%	55105
f_4	100%	57400	100%	35731
f_5	100%	67695	100%	28784
f_6	100%	21993	100%	14230
f_7	0%	N/A	100%	755490
f_7^*	0%	N/A	N/A	N/A
f_8	0%	N/A	100%	230411
f_8^*	0%	N/A	N/A	N/A
f_9	100%	78754	100%	48521
f_{10}	0%	N/A	23%	896867
f_{10}^*	1%	878400	N/A	N/A
f_{11}^{**}	100%	371211	100%	366342

In the tables the star sign (*) in the first column signifies that the test was repeated with increased tolerance ($tolerance = 1\%$), while the double star sign (**) signifies that a different value for parameter N was used (it applied only to f_{11} and it was set to $N = 200$ for $n = 10, 20$, and $N = 400$ for $n = 30$). The abbreviation "N/A" in Tables 6, 7 and 8 has the meaning of "not applicable" and it is used for $\mu(NFE)$ when the optimization problem cannot be solved ($SR\%$ is 0), and also for both $\mu(NFE)$ and $SR\%$ when a test is repeated with an increased tolerance for the method which is not able to solve the optimization problem with the usual tolerance (0.1%), but it is not repeated for the method which provides good results with the usual tolerance, because it is not considered of interest (*Note*: when $SR\%$ is "N/A" it does not mean that the optimization problem cannot be solved, quite the opposite, with a more relaxed tolerance the $SR\%$ would be expected to increase).

Table 6 presents the comparative testing results, obtained for $n = 10$, between the final *FSA-DE* method and the initial scheme *DE/rand/1/bin*. It can be observed that for *FSA-DE* the success rate was notably increased for f_{10}, while for the other test functions both methods produced excellent results. The efficiency increased for all the test functions, notably for f_1, f_4, f_7, f_8 and f_{10}.

Table 7 presents the comparative testing results, obtained for $n = 20$, between the final *FSA-DE* method and the initial scheme *DE/rand/1/bin*. From the success rate perspective *DE/rand/1/bin* started having problems for f_8, had serious problems for f_{10}, and was not able to solve f_7, while *FSA-DE* produced excellent results for all test functions with the exception of f_{10}. From the efficiency perspective *FSA-DE* was clearly better than *DE/rand/1/bin* for all the test functions that the later was also able to solve,

with the exception of f_{11} where the difference was small.

Table 8 presents the comparative testing results, obtained for $n = 30$, between the final *FSA-DE* method and the initial scheme *DE/rand/1/bin*. From the success rate perspective *DE/rand/1/bin* had serious problems for f_1, was not able to solve f_7, f_8 and f_{10}, but was able to solve the other problems with maximum success rate percentages, while *FSA-DE* had serious problems with f_{10}, but for the other test functions produced the maximum success rate percentages. From the efficiency perspective *FSA-DE* is clearly better for all the test functions that *DE/rand/1/bin* was also able to solve with the exception of f_{11}, where the difference was small.

As a general conclusion, with the increase of the search space dimension, *FSA-DE* was able to solve all the test functions, while *DE/rand/1/bin* was not able to solve some of the test functions. Both methods were able to produce excellent success rates for the test functions with a regular structure of the modes (the modes arranged in a lattice structure parallel with the coordinate system), while the improved *FSA-DE* method clearly provided better success rates for test functions not presenting regularly structured modes. The improved *FSA-DE* method was consistently more efficient than *DE/rand/1/bin* for all test functions and in some cases the improvement was notable.

12 Conclusions

In the paper was built an improved variant of the *DE* scheme, named Fast Self-Adaptive DE (*FSA-DE*), by implementing and testing a set of gradual and cumulative improvements to the initial *DE/rand/1/bin* scheme with fixed control parameters, $F = 0.5$ and $C_r = 0.5$. The purpose was to finally obtain an improved optimization method in both success rate and efficiency, while eliminating the dependence on the control parameters F and C_r. For testing purposes a set of 11 scalable, multimodal, continuous optimization functions (10 unconstrained and 1 constrained) with known global solutions was selected. The tested modifications consisted in the randomization of the F control parameter (a simple jitter scheme), a more efficient Random Greedy Selection scheme, an adaptive scheme for the C_r control parameter and a resetting scheme for the agents' positions. The testing was conducted by employing a testing methodology which emphasizes the importance of both success rate and efficiency. Finally there was investigated the performance degradation of the compared optimization methods with the increase of the search space dimension. The novel *FSA-DE* method was able

to solve all the test functions for all the considered search space dimensions in the given test conditions. The main differences were observed for test functions not presenting regularly structured modes, for which the initial scheme presented serious difficulties with the increase of the search space dimension, while the novel method was able to solve all of them with an overall good performance.

References

[1] **K. Price and R. Storn**, Differential Evolution - A simple and efficient adaptive scheme for global optimization over continuous spaces, *Journal of Global Optimization*, **11** (4), (1997), 341–359

[2] **R. Storn**, On the usage of differential evolution for function optimization, *Biennial Conference of the North American Fuzzy Information Processing Society (NAFIPS)*, (1996), 519–523

[3] **K. Price, R. Storn, and J. Lampinen**, *Differential Evolution - A Practical Approach to Global Optimization*, Springer-Verlag, Berlin, Heidelberg, 2005

[4] **J. Liu and J. Lampinen**, On setting the control parameter of the differential evolution method, *Proceedings of the 8th International Conference on Soft Computing (MENDEL)*, (2002), 11–18

[5] **D. Zaharie**, Critical values for the control parameters of differential evolution algorithms, *Proceedings of the 8th International Conference on Soft Computing (MENDEL)*, (2002), 62–67

[6] **J. Liu and J. Lampinen**, A fuzzy adaptive differential evolution algorithm, *Soft Computing*, **9** (6), (2005), 448–462

[7] **A.K. Qin and P.N. Suganthan**, Self-adaptive differential evolution algorithm for numerical optimization, *In Proceedings of the IEEE Congress on Evolutionary Computation (CEC), 2005*, **2**, (2005), 1785–1791

[8] **A.K. Qin, V.L. Huang, and P.N. Suganthan**, Differential evolution algorithm with strategy adaptation for global numerical optimization, *IEEE Transactions on Evolutionary Computation*, **13** (2), (2008), 398–417

[9] **J. Brest, S. Greiner, B. Bošković, M. Mernik, and V. Zumer**, Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark functions, *IEEE Transactions on Evolutionary Computation*, **10** (6), (2006), 646–657

[10] **S. Das and P.N. Suganthan**, Differential Evolution: A Survey of the State-of-the-Art, *IEEE Transactions on Evolutionary Computation*, **15** (1), (2011), 4–31

[11] **S. Das, P.N. Suganthan, and S.S. Mullick**, Recent advances in differential evolution - An updated survey, *Swarm and Evolutionary Computation*, **27** (1), (2016), 1–30

[12] **F. Neri and V. Tirronen**, Recent advances in differential evolution: a survey and experimental analysis, *Artificial Intelligence Review*, **33** (1-2), (2010), 61–106

[13] **J.D. Pintér**, *Global Optimization: Software, Test Problems, and Applications, Ch. 15 in Handbook of Global Optimization, Volume 2, (Ed. P. M. Pardalos and H. F. Romeijn)*, Kluwer Academic Publishers, Dordrecht, Boston, London, 2002

[14] **K. Deb**, An efficient constraint handling method for genetic algorithms, *Computer Methods in Applied Mechanics and Engineering*, **186** (2-4), (2000), 311–338

[15] **R. Mallipeddi and P.N. Suganthan**, Differential Evolution with Ensemble of Constraint Handling Techniques for solving CEC 2010 Benchmark Problems, *In Proceedings IEEE Congress on Evolutionary Computation (CEC), 2010*, 1–8

[16] **G. Anescu and I. Prisecaru**, NSC-PSO, a novel PSO variant without speeds and coefficients, *Proceedings of 17th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC 2015*, 460–467

[17] **G. Anescu**, An imperialistic strategy approach to continuous global optimization problem, *Proceedings of 16th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC 2014*, 549–556

[18] **Z. Michalewicz**, *Genetic Algorithms + Data structures = Evolution Programs*, Springer-Verlag, New York, 1994

[19] **J. Momin and Yang Xin-She**, A literature survey of benchmark functions for global optimization problems, *Int. Journal of Mathematical Modelling and Numerical Optimisation*, **4** (2), (2013), 150–194

[20] **M. Molga and C. Smutnicki**, Test functions for optimization needs, *http://www.robertmarks.org/Classes/ENGR5358/Papers/functions.pdf (last time accessed in February, 2016)*, (2005), 1–43

[21] **A.J. Keane**, Experiences with optimizers in structural design, *Proceedings of the 1st Conf. on Adaptive Computing in Engineering Design and Control, University of Plymouth, UK*, (1994), 14–27

[22] **S.K. Mishra**, Minimization of Keane's Bump Function by the Repulsive Particle Swarm and the Differential Evolution Methods, *http://mpra.ub.uni-muenchen.de/3098/ (last time accessed in February, 2016)*, (2007), 1–12

[23] **R. Storn**, Optimization of wireless communications applications using differential evolution, *In SDR Technical Conference, Denver*, (2007)

[24] **D. Karaboga and B. Basturk**, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, *Journal of Global Optimization*, **39** (3), (2007), 459–471

George Anescu

Power Engineering Faculty, Polytechnic University of Bucharest
313 Splaiul Independentei, 060042, Bucharest, Romania

E-mail: `george.anescu@gmail.com`

Received: 23.09.2016

Accepted: 2.12.2016