Proceedings of the International Multiconference on Computer Science and Information Technology pp. 927–931
ISBN 978-83-60810-14-9
ISSN 1896-7094

Adaptive Differential Evolution and Exponential Crossover

Josef Tvrdík
University of Ostrava, Department of Computer Science
Email: josef.tvrdik@osu.cz

***Abstract*—Several adaptive variants of differential evolution are described and compared in two sets of benchmark problems. The influence of exponential crossover on efficiency of the search is studied. The use of both types of crossover together makes the algorithms more robust. Such algorithms are convenient for the real-world problems, where we need an adaptive algorithm applicable without time-wasting parameter tuning.**

I. Introduction

THE GLOBAL optimization problem is considered in this paper in the following form:

$$\text{minimize } f(\boldsymbol{x}) \quad \text{subject to} \quad \boldsymbol{x} \in D,$$

where $\boldsymbol{x}$ is a continuous variable with the domain $D \subset \mathbb{R}^d$, and $f(\boldsymbol{x}) : D \to \mathbb{R}$ is a continuous function. The domain D is defined by specifying boundary constrains, that are lower (a_j) and upper (b_j) limits of each component j, $D = \prod_{j=1}^{d}[a_j, b_j],\ a_j < b_j,\ j = 1, 2, \ldots, d$. The global minimum point $\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}\in D} f(\boldsymbol{x})$ is the solution of the problem. The global optimization problems that use boundary constrains only are usually referred as unconstrained ones [1], due to the fact, that boundary constrains are easy to handle and in computer numerical solution there are "natural" boundary constrains, given by the representation of numbers in floating-point format.

The global optimization problem is not easy to solve and standard deterministic algorithms tend to stop the search in local minimum nearest to the input starting point. Therefore, heuristic search is widely used in the global optimization. Such heuristics are often inspired by natural processes, mainly by the evolution in populations. Evolutionary algorithms are able to find the acceptable solution with reasonable time demand, but efficiency of the search is sensitive to the setting of control parameters. Application of evolutionary algorithms usually requires time-consuming tuning of control parameters to the problem to be solved. Thus, great effort has been focused to the development of self-adaptive variants of evolutionary algorithms, applicable without preliminary tuning of control parameter.

II. Differential Evolution

The differential evolution (DE) was introduced by Storn and Price [2]. Nowadays the DE has become one of the most frequently used evolutionary algorithms solving the global optimization problems [3]. The algorithm of DE in pseudo-code is shown in Fig. 1. New trial point $\boldsymbol{y}$ (line 4 in DE algorithm) is generated by using mutation and crossover.

There are various strategies, how to create the mutant point $\boldsymbol{v}$. The most popular strategy denoted by abbreviation DE/rand/1/ generates the point $\boldsymbol{v}$ by adding the weighted difference of two points

$$\boldsymbol{v} = \boldsymbol{r}_1 + F\,(\boldsymbol{r}_2 - \boldsymbol{r}_3)\,, \tag{1}$$

where $\boldsymbol{r}_1, \boldsymbol{r}_2$ and $\boldsymbol{r}_3$ are three mutually distinct points taken randomly from population P, not coinciding with the current $\boldsymbol{x}_i$, and $F > 0$ is input parameter.

```
1 initialize population P = (x_1, x_2, ..., x_N), x_i ∈ D
2 repeat
3     for i := 1 to N do
4         generate a new trial vector y
5         if f(y) < f(x_i) then insert y into new generation Q
6                          else insert x_i into new generation Q
7         endif
8     endfor
9     P := Q
10 until stopping condition
```

Fig. 1. Differential evolution—Algorithm in pseudo code

Kaelo and Ali [4] proposed a slightly different attempt to generating a mutant point $\boldsymbol{v}$ by (1). They called it random localization. The point $\boldsymbol{r}_1$ is not chosen randomly, but it is tournament best among $\boldsymbol{r}_1$, $\boldsymbol{r}_2$, and $\boldsymbol{r}_3$, that is $\boldsymbol{r}_1 = \arg\min_{i\in\{1,2,3\}} f(\boldsymbol{r}_i)$.

The crossover operator constructs the offspring $\boldsymbol{y}$ by mixing components of the current individual $\boldsymbol{x}_i$ and the point $\boldsymbol{v}$ generated by mutation. There are two types of crossover used in DE, binomial and exponential ones.

Binomial crossover replaces the elements of vector $\boldsymbol{x}_i$ using the following rule

$$y_j = \begin{cases} v_j & \text{if} \quad U_j \le CR \quad \text{or} \quad j = l \\ x_{ij} & \text{if} \quad U_j > CR \quad \text{and} \quad j \ne l\,, \end{cases} \tag{2}$$

where l is a randomly chosen integer from $\{1, 2, \ldots, d\}$, and $U_1, U_2, \ldots, U_d$ are independent random variables uniformly distributed in $[0,\ 1)$. $CR \in [0, 1]$ is an input parameter influencing the number of elements to be exchanged by crossover. Eq. (2) ensures that at least one element of $\boldsymbol{x}_i$ is changed,

even if $CR = 0$. This kind of crossover according to (2) is commonly expressed by abbreviation DE/·/·/bin.

For exponential crossover (DE/·/·/exp), the starting position of crossover is chosen randomly from $1, \ldots, d$, and L consecutive elements (counted in circular manner) are taken from the mutant vector $\boldsymbol{v}$. Probability of replacing the k-th element in the sequence $1, 2, \ldots, L, \quad L \leq d$, decreases exponentially with increasing k. There are two main differences between binomial and exponential crossovers:

- L adjacent elements are changed in exponential variant, in binomial one the changed coordinates are dispersed randomly over the dimensions $1, 2, \ldots, d$.
- While the relation between the probability of crossover and the *CR* is linear in binomial crossover; in the exponential one, this relation is nonlinear, and the deviation from linearity enlarges with increasing dimension of problem.

Probability of crossover, p_m, determines the number of exchanged elements (p_m being the mean value of relative frequency). Zaharie [5] derived the relation between p_m and *CR* for exponential crossover. Her result can be rewritten in the form of polynom

$$CR^d - \ d\ p_m\ CR + \ d\ p_m - 1 = 0. \tag{3}$$

It is apparent that the polynom (3) has the only one real root in the open interval of $(0, 1)$ for $p_m \in (1/d, 1)$. The crossover parameter $CR = 0$ for $p_m = 1/d$, and for $p_m = 1$ the value of $CR = 1$. Thus, for given p_m we can find unique corresponding value of *CR*, to be set as crossover parameter.

Differential evolution has a few control parameters only, namely the size of population N, selection of mutation strategy, choice of crossover type, and pair of parameters F and *CR*. However, the efficiency of differential evolution is very sensitive especially regarding the setting of F and *CR* values. The most suitable control parameters values for specific problem may be found by trial-and-error tuning, but it requires a lot of time. There are some recommendations for the setting of these parameters. Zaharie [6] suggested the intervals for the control parameters based on the variability of population, for other proposals see [2], [3], [7], [8].

Self-adaptive setting of control parameters was studied in several papers. Ali and Törn [9] proposed a simple rule for adapting the F scaling factor value during the search process. Some other attempts to the adaptation of DE control parameters are summarized in Liu and Lampinen [10]. Recently, Quin and Suganthan [11] proposed self-adaptive choice of mutation strategy combined with controlled random adjusting the values of F and *CR*.

Evolutionary self-adaptation of control parameters F and *CR* suggested by Brest et al. [12] and competitive self-adaptive setting of the control parameters introduced by Tvrdík [13] (both variants with binomial crossover) have proved good convergence in the applications to various problems of the global optimization [14]. Experimental comparison of these self-adaptive patterns and influence of exponential crossover is the goal of this paper. These self-adaptive algorithms are described in detail in next section.

III. Variants of DE in Comparison

DE with evolutionary self-adaptation of F and *CR* was proposed by Brest et al. [12]. The values of F and *CR* are initialized randomly for each point in population and survive with the individuals in population, but they can be mutated randomly in each generation with given probabilities τ_1, τ_2. Authors used binomial crossover with the values of $CR \in [0, 1]$ distributed uniformly, and DE/rand/1 mutation with values of F also distributed uniformly in $[F_l, F_u]$. They set input parameters $F_l = 0.1$, and $F_u = 0.9$.

The competitive setting of the control parameters was described in [13]. In this self-adaptation approach, we choose among H different settings of F and *CR* randomly with probabilities $q_h, \ h = 1, 2, \ldots, H$. These probabilities are modified according to the success rate of the settings in preceding steps of search process. The h-th setting is considered successful, if it generates such a trial point $\boldsymbol{y}$ that $f(\boldsymbol{y}) < f(\boldsymbol{x}_i)$. Probability q_h is evaluated as the relative frequency

$$q_h = \frac{n_h + n_0}{\sum_{j=1}^{H}(n_j + n_0)}\ , \tag{4}$$

where n_h is the current count of the h-th setting successes, and $n_0 > 0$ is a constant. The setting of $n_0 > 1$ prevents a dramatic change in q_h by one random successful use of the h-th parameter setting. To avoid degeneration of the search process, the current values of q_h are reset to their starting values $q_h = 1/H$, if any probability q_h decreases bellow some given limit $\delta > 0$. Several variants of such competitive differential evolution with binomial crossover were numerically compared in [15], and two best performing variants were included into open Matlab library [16].

The mutation according to (1) could cause that a new trial point $\boldsymbol{y}$ moves out of the domain D. In such case, the point $\boldsymbol{y}$ can be skipped and a new one generated. More effective attempt is to replace value of $y_j < a_j$ or $y_j > b_j$, either by random value in $[a_j, b_j]$, see [7], or by reversed value of this coordinate y_j into D over the a_j or b_j by mirroring, see [17]. The latter method is used in algorithms implemented for numerical tests.

The goal of experiments was to compare the impact of crossover type to the performance of in the both self-adaptive kinds of DE algorithm.

The four most efficient competitive variants from twelve variants of DE tested in [18] are selected for experimental comparison. These variants have proved significantly better efficiency than the standard DE. Two variants in experiments are based on the Brest approach to self-adaptation [12] and two other variants in experimental tests combine both approaches. The tested variants are denoted by mnemonic labels.

The labels of competitive variants begin with capital letter "C" in bellow text. In variant label, the type of crossover is followed by number of competing settings. The presence of random localization is marked by suffix "rl". Where the

variant uses both types of crossover, "rl" is related to both of them. Nine settings of F and *CR* for binomial crossover are created from all combinations of $F \in \{0.5, 0.8, 1\}$ and $CR \in \{0, 0.5, 1\}$). If there are 6 settings in competition only, the value $F = 1$ is not used. The same values of F are used for exponential crossover. However, *CR* values are derived from the crossover probability, given by (3). Actual values of crossover probability can vary in interval $[1/d,\ 1]$, because at least one coordinate of current point $\boldsymbol{x}_i$ in changed in both types of crossover. Notice, that for extreme values of probability, $p_m = 1/d$ and $p_m = 1$, exponential crossover does not differ from binomial one. In order to ensure the different behaviour of exponential crossover from binomial one, the following three levels of crossover probability p_1, p_2, p_3 were chosen. Value of p_2 is in the middle of the interval $[1/d,\ 1]$, p_1 in the middle of the interval $[1/d,\ p_2]$, and p_3 in the middle of the interval $[p_2,\ 1]$, i.e.

$$p_2 = \frac{1 + 1/d}{2}, \quad p_1 = \frac{1/d + p_2}{2}, \quad p_3 = \frac{p_2 + 1}{2}. \tag{5}$$

The values of crossover probability and the corresponding values of parameter *CR* for $d = 10$ and $d = 30$ are given in Table I.

TABLE I
VALUES OF CROSSOVER PROBABILITY AND THE CORRESPONDING VALUES OF *CR* PARAMETER FOR EXPONENTIAL CROSSOVER

	$d = 10$				$d = 30$		
i	1	2	3		1	2	3
p_i	0.3250	0.5500	0.7750	p_i	0.2750	0.5167	0.7583
CR_i	0.7011	0.8571	0.9418	CR_i	0.8815	0.9488	0.9801

Two variants based on Brest approach are labeled *Brestbin* and *Brestexp*. *Brestbin* uses binomial crossover with control parameters values recommended in [12], i.e. $\tau_1 = \tau_2 = 0.1$ and $F_l = 0.1$, $F_u = 0.9$. In the *Brestexp* variant, values of *CR* are distributed uniformly in $[CR_1, CR_3]$, see Table I. The random localization was not applied in these variants.

The last two variants of DE in tests combine both the competitive and Brest self-adaptive mechanisms. Two types of crossover compete according the rules described above, the parameters of crossover are the same as in *Brestbin* and *Brestexp*. Two pairs of *CR* and F values are kept with each individual in population, the first pair is used for exponential crossover and the second pair for binomial one. Variant labeled *Brestcmp* is without random localization. In variant labeled *Brestcmprl* random localization is applied to the mutation.

Thus, four competitive DE variants, labeled *Cbin9rl*, *Cexp9rl*, *Cbin9exp9rl*, and *Cbin6exp6rl* are included into experimental comparison with four other variants labeled *Brestbin*, *Brestexp*, *Brestcmp*, and *Brestcmprl*.

IV. EXPERIMENTS AND RESULTS

A. Standard Benchmark

1) Test Functions: Five commonly used functions [9], [3], [2] were applied as standard benchmark:

- Ackley function—multimodal, separable,

$$f(\boldsymbol{x}) = -20 \exp\left(-0.02\ \sqrt{\tfrac{1}{d}\sum_{j=1}^{d} x_j^2}\right) - \\ - \exp\left(\tfrac{1}{d}\sum_{j=1}^{d} \cos 2\pi x_j\right) + 20 + \exp(1)$$

  $x_j \in [-30, 30]$, $\boldsymbol{x}^* = (0, 0, \ldots, 0)$, $f(\boldsymbol{x}^*) = 0$.
- Griewank function—multimodal, nonseparable,

$$f(\boldsymbol{x}) = \sum_{j=1}^{d} \frac{{x_j}^2}{4000} - \prod_{j=1}^{d} \cos\left(\frac{x_j}{\sqrt{j}}\right) + 1$$

  $x_j \in [-400, 400]$, $\boldsymbol{x}^* = (0, 0, \ldots, 0)$, $f(\boldsymbol{x}^*) = 0$.
- Rastrigin function—multimodal, separable,

$$f(\boldsymbol{x}) = 10\,d + \sum_{j=1}^{d} [{x_j}^2 - 10\cos(2\pi x_j)]$$

  $x_j \in [-5.12, 5.12]$, $\boldsymbol{x}^* = (0, 0, \ldots, 0)$, $f(\boldsymbol{x}^*) = 0$
- Rosenbrock function (banana valley)—unimodal, nonseparable,

$$f(\boldsymbol{x}) = \sum_{j=1}^{d-1} \left[100(x_j^2 - x_{j+1})^2 + (1 - x_j)^2\right]$$

  $x_j \in [-2.048, 2.048]$, $\boldsymbol{x}^* = (1, 1, \ldots, 1)$, $f(\boldsymbol{x}^*) = 0$.
- Schwefel function—multimodal, separable, the global minimum is distant from the next best local minima,

$$f(\boldsymbol{x}) = -\sum_{j=1}^{d} x_j \sin(\sqrt{|\ x_j\ |})$$

  $x_j \in [-500, 500]$, $\boldsymbol{x}^* = (s, s, \ldots, s)$, $s = 420.968746$, $f(\boldsymbol{x}^*) = -418.982887\,d$

Two levels of problem dimension were chosen, $d = 10$ and $d = 30$. The names of test functions (or their self-explaining abbreviations) are used as labels when reporting the results.

2) Experiments and Results: One hundred of independent runs were carried out for each function and level of d. The search was stopped if the difference between maximum and minimum function values was less than $1e{-}6$ or number of function evaluations (hereinafter *ne*) exceeded the limit $20000\,d$. Population size was set to 40 for problems with $d = 10$ and 60 for for problems with $d = 30$. Parameters controlling the competition of settings were set to $n_0 = 2$, and $\delta = 1/(5\,H)$.

Number of function evaluations *ne* and the success of search were recorded in each run. The search was considered successful, if $f_{min} - f(\boldsymbol{x}^*) < 1e - 4$, where f_{min} is the minimum value of objective function in final population. The reliability (estimator of convergence probability), hereinafter R, was evaluated as the percentage of successful runs.

The performance of variants were compared using Q-measure. The Q-measure was proposed by Feoktistov [7] to integrate time demand and reliability into a single criterion of convergence. The formula of Q-measure is

$$Q_m = \overline{ne}/R, \tag{6}$$

where $\overline{ne}$ is the average of number of function evaluations in successful runs and R is the reliability in %.

TABLE II
COMPARISON OF VARIANTS USING THE VALUES OF Q-MEASURE.

d	Variant	Ackley	Griewank	Rastrig	Rosen	Schwefel	Average
10	Cbin9rl	176	179	**152**	**250**	135	178
10	Cexp9rl	188	183	198	286	142	199
10	Cbin9exp9rl	181	183	165	**245**	140	183
10	Cbin6exp6rl	157	**156**	**150**	272	**122**	**172**
10	Brestbin	152	161	198	717	125	270
10	Brestexp	152	169	192	680	126	264
10	Brestcmp	**148**	157	191	852	123	294
10	Brestcmprl	**108**	**152**	157	308	**96**	**164**
30	Cbin9rl	911	688	**793**	2542	746	1136
30	Cexp9rl	822	626	1504	**1511**	848	1062
30	Cbin9exp9rl	898	678	846	1756	758	987
30	Cbin6exp6rl	763	574	**781**	1601	661	**876**
30	Brestbin	583	477	1596	3777	607	1408
30	Brestexp	574	525	1393	1953	632	1015
30	Brestcmp	**557**	**453**	1431	2226	**587**	1051
30	Brestcmprl	**434**	**420**	1131	**1455**	**494**	**787**

Best and second best values of Q_M are bold, the best values are underlined.

TABLE III
RELIABILITY OF THE SEARCH—VALUES OF R.

d	Variant	Ackley	Griewank	Rastrig	Rosen	Schwefel	Average
10	Cbin9rl	100	100	100	100	100	100.0
10	Cexp9Rrl	100	100	100	100	100	100.0
10	Cbin9exp9rl	100	99	100	100	99	99.6
10	Cbin6exp6rl	100	100	100	100	100	100.0
10	Brestbin	100	96	99	98	99	98.4
10	Brestexp	100	89	100	100	100	97.8
10	Brestcmp	100	96	100	100	99	99.0
10	Brestcmprl	99	64	95	96	93	89.4
30	Cbin9rl	100	100	100	100	100	100.0
30	Cexp9rl	100	100	100	100	100	100.0
30	Cbin9exp9rl	100	100	100	100	100	100.0
30	Cbin6exp6rl	100	100	100	100	100	100.0
30	Brestbin	100	97	100	100	98	99.0
30	Brestexp	100	82	100	98	98	95.6
30	Brestcmp	100	95	100	100	100	99.0
30	Brestcmprl	85	66	100	93	82	85.2

The values of Q-measure for all variants and test functions are shown in Table II. The best and second best values of Q_m are printed bold, the best values are underlined. Comparing the performance of self-adaptive variants of DE, *Brestcmprl* was the best in 7 of 10 test problems and its averages of Q_m are the smallest for both dimension levels, but it does not appeared within best two variants in some problems (Rosenbrock, problem dimension $d = 10$, Rastrigin, both levels of dimension).

Reliability of the search is shown in Table III. Comparing the values of Q_m in Table II and reliability in Table III, it is apparent, that higher efficiency of *Brestcmprl* is paid by less reliability of the search by this algorithm. Overall view shows that competitive variants are more reliable, but sometimes less efficient than the variants based on the evolutionary self-adaptive mechanism proposed by Brest et al.

B. Composition Functions

The novel hybrid benchmark functions were proposed by Liang et al. [19], in order to make testing more similar to real-world problems. Commonly used standard benchmark problems are criticized due to the fact that their global minimum points have the same coordinate values in each dimension (the case of all functions in our standard benchmark), or the global minimum point is in the domain center of gravity (the case of Ackley, Griewank, and Rastrigin). The novel hybrid benchmark functions are not featured with any kind of such symmetry. Each of them is composed from several various functions by non-trivial transformations including matrix rotation. Thus, they are non-separable functions with many local minima, and the global minimum point cannot be found easily.

Six evolutionary algorithms were tested in this composite functions benchmark in [19]. Among these algorithms, comprehensive learning particle swarm optimization (CLPSO) was the most efficient in CF1, CF2, CF3, and CF5 problems, standard PSO algorithm in CF4 problem, and a variant of DE with exponential crossover in CF6 problem. Details and references are given in Liang et al.

1) Experiments and Results: The same eight variants of self-adaptive DE tested with standard benchmark were applied to hybrid benchmark functions. To enable the comparison of our results with the experimental results of the algorithms in [19], the arrangement of experiments was the same, i.e. for each test function, each algorithm was run 20 times and maximum function evaluations are set to 50,000 for all the algorithms. The domains are $D = \prod_{j=1}^{10}[-5, 5]$ for all test functions. Population size was set up to $N = 50$ for all the algorithms in tests. Other control parameters had values used for standard benchmark.

The averages and standard deviations of computed minimum function values of 20 runs are reported in Table IV respectively. Results of the best algorithm from [19] are reported, as well. The values less than the best ones found in [19] are printed bold, the least value for each function is underlined. The value 0 in CF1 column stands for the positive values less than 4.94e-324, which is the accuracy of double-precision arithmetic used in the implementation environment.

From eight new self-adaptive variants of DE, at least four variants outperformed the most successful algorithm in [19] for each function except CF6. In the case of CF4 even all seven variants were more efficient than the best algorithm reported in [19]. For CF1 problem, five self-adaptive variants achieved better results than those reported former, but the solution found by CLPSO in [19] is acceptable, so the improvement attained by self-adaptive variants is not important. Surprisingly, the bold values of standard deviation in Table IV are scarce, what implies that only small part of self-adaptive DE variants is more robust than the best algorithms from [19].

There is no unique winner among the DE variants tested on composition benchmark problems. However, we can con-

TABLE IV
RESULTS ACHIEVED BY THE ALGORITHMS ON SIX COMPOSITION FUNCTIONS

Averages of computed minimum function values in 20 runs						
Algorithm	CF1	CF2	CF3	CF4	CF5	CF6
Best in [19]	5.7e-08	19.2	133	314	5.37	491
Cbin9rl	**0**	**5.7**	**103**	**285**	**5.10**	560
Cexp9rl	**0**	21.6	**108**	**305**	**0.06**	585
Cbin9exp9rl	**0**	21.0	**100**	**285**	10.27	580
Cbin6exp6rl	10	**15.4**	**111**	**296**	**5.13**	656
Brestbin	5	**10.3**	133	**274**	**0.06**	535
Brestexp	**0**	**10.8**	**104**	**270**	10.47	540
Brestcmp	5	**6.1**	**106**	**271**	10.24	500
Brestcmprl	20	31.9	**112**	**285**	15.68	596
Standard deviations of computed minimum function values in 20 runs						
Algorithm	CF1	CF2	CF3	CF4	CF5	CF6
Best in [19]	1.0e-07	14.7	20.0	20.1	2.61	39.5
Cbin9rl	**0**	22.2	32.8	**13.7**	22.34	147.4
Cexp9rl	**0**	51.7	47.6	71.5	**0.25**	149.5
Cbin9exp9rl	**0**	51.9	35.4	**9.3**	30.69	165.1
Cbin6exp6rl	30.8	36.5	39.6	72.4	22.33	207.6
Brestbin	22.4	30.7	35.9	**13.3**	**0.29**	127.5
Brestexp	**0**	30.5	24.5	23.6	30.63	123.7
Brestcmp	22.4	22.1	41.5	**14.8**	30.70	**0.1**
Brestcmprl	52.3	56.4	28.5	**11.1**	36.35	183.0

clude that the novel self-adaptive DE variants were able to outperform advanced evolutionary algorithms in the majority of these hard test problems.

V. CONCLUSIONS

Both self-adaptive patterns used in tested variants have proved the efficiency. All eight novel self-adaptive DE variants outperformed standard evolutionary algorithms in some test problems.

These results were achieved with implicit setting of control parameters without tuning their values. For given subset of benchmark problems the population size was set the same for all the DE variant. The influence of population size on algorithm efficiency was not examined in this study. This question should be solved in future including an attempt to solve some self-adaptive mechanism of population size.

The use of stand-alone exponential crossover increases efficiency in comparison with binomial crossover only for small part of problems. Applying competitive parameter setting together with both crossover types brings better convergence in standard benchmark problems. The combination of both self-adapting approaches results in the most efficient algorithm for standard benchmark problems. In the case of composition benchmark, the combination of both crossover types was beneficial for less part of problems.

Summarizing the results from both benchmarks, there is no generally winning variant. This is an implication resulting from the No Free Lunch Theorems [20], stating that any stochastic search algorithm cannot outperform others, when all possible objective functions are taken into account. Nevertheless, it does not exclude the possibility to propose novel variants of self-adaptive algorithms with better efficiency for wider range of objective functions and the self-adaptive variants of DE seem to be good examples of such algorithms advisable to the application in real-world problems of global optimization.

ACKNOWLEDGMENT

Supported by the grant 201/08/0472 of the Czech Grant Agency and by the research project MSM 6198898701.

REFERENCES

[1] A. P. Engelbrecht, *Computatinal Intelligence: An Introduction*. Chichester: John Wiley & sons, 2007.
[2] R. Storn and K. V. Price, "Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optimization*, vol. 11, pp. 341–359, 1997.
[3] K. V. Price, R. Storn, and J. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*. Springer, 2005.
[4] P. Kaelo and M. M. Ali, "A numerical study of some modified differential evolution algorithms," *European J. Operational Research*, vol. 169, pp. 1176–1184, 2006.
[5] D. Zaharie, "A comparative analysis of crossover variants in differential evolution," in *Proceedings of IMCSIT 2007*, U. Markowska-Kaczmar and H. Kwasnicka, Eds. Wisla: PTI, 2007, pp. 171–181.
[6] ——, "Critical values for the control parameter of the differential evolution algorithms," in *MENDEL 2002, 8th International Conference on Soft Computing*, R. Matoušek and P. Ošmera, Eds. Brno: University of Technology, 2002, pp. 62–67.
[7] V. Feoktistov, *Differential Evolution in Search of Sotution*. Springer, 2006.
[8] R. Gämperle, S. D. Müller, and P. Koumoutsakos, "A parameter study for differential evolution," in *Advances in Intelligent Systems Fuzzy Systems, Evolutionary Computing*, A. Grmela and N. E. Mastorakis, Eds. Athens: WSEAS Press, 2002, pp. 293–298.
[9] M. M. Ali and A. Törn, "Population set based global optimization algorithms: Some modifications and numerical studies," *Computers and Operations Research*, vol. 31, pp. 1703–1725, 2004.
[10] J. Liu and J. Lampinen, "A fuzzy adaptive differential evolution algortithm," *Soft Computing*, vol. 9, pp. 448–462, 2005.
[11] A. K. Quin and P. N. Suganthan, "Self-adaptive differential evolution for numerical optimization," in *IEEE Congress on Evolutionary Computation*, 2005, pp. 1785–1791.
[12] J. Brest, S. Greiner, B. Boškovič, M. Mernik, and V. Žumer, "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, pp. 646–657, 2006.
[13] J. Tvrdík, "Competitive differential evolution," in *MENDEL 2006, 12th International Conference on Soft Computing*, R. Matoušek and P. Ošmera, Eds. Brno: University of Technology, 2006, pp. 7–12.
[14] ——, "Adaptation in differential evolution: A numerical comparison," *Applied Soft Computing*, 2007, submitted.
[15] ——, "Differential evolution with competitive setting of its control parameters," *TASK Quarterly*, vol. 11, pp. 169–179, 2007.
[16] J. Tvrdík, V. Pavliska, and H. Habiballa, "Stochastic algorithms for global optimization—matlab and c++ library," University of Ostrava, 2007. [Online]. Available: http://albert.osu.cz/oukip/optimization/
[17] V. Kvasnička, J. Pospíchal, and P. Tiňo, *Evolutionary Algorithms*. Bratislava: Slovak Technical University, 2000, (In Slovak).
[18] J. Tvrdík, "Exponential crossover in competitive differential evolution," in *MENDEL 2008, 14th International Conference on Soft Computing*, R. Matoušek, Ed. Brno: University of Technology, 2008, pp. 44–49.
[19] J. J. Liang, P. N. Suganthan, and K. Deb, "Novel composition test functions for numerical global optimization," in *Proc. IEEE Swarm Intelligence Symposium*, 2005, pp. 68–75, matlab codes on web. [Online]. Available: http://www.ntu.edu.sg/home/EPNSugan/index_files/comp-functions.htm
[20] D. H. Wolpert and W. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, pp. 67–82, 1997.