Competitive Differential Evolution Algorithm in Comparison with Other Adaptive Variants

Radka Poláková and Josef Tvrdík

Centre of Excellence IT4Innovations division of University of Ostrava
Institute for Research and Applications of Fuzzy Modeling
{radka.polakova,josef.tvrdik}@osu.cz

Abstract. The differential evolution algorithm using competitive adaptation was compared experimentally with the state-of-the-art adaptive versions of differential evolution on CEC2005 benchmark functions. The results of experiments show that the performance of the algorithm with competitive adaptation is comparable with the state-of-the-art algorithms, outperformed only by CoDE and JADE algorithms in this test. A modification of competitive differential evolution preferring successful strategy for a longer period of search was also investigated. Such modification brings no improvement and the standard setting of the competition recommended in previous papers is suitable for applications.

1 Introduction

Differential evolution (DE) is simple population-based algorithm for the global optimization introduced by Storn and Price [8]. DE has become one of the evolutionary algorithms most frequently used for solving the global optimization problems in recent years [6]. Compared with other evolutionary algorithms, DE has a very small number of control parameters. However, it is commonly known that the performance of DE algorithm is strongly dependent on the values of these parameters. Tuning the proper values of control parameters for solving a particular optimization problem by trial-and-error can take a lot of time. Because of this fact many adaptive approaches in DE have appeared in literature. Four of them, namely jDE [1], SADE [7], JADE [16], and EPSDE [4] are usually considered as the state-of-the-art adaptive variants of DE algorithm. DE algorithm with composite trial vector generation strategies and control parameters (CoDE) has appeared recently and its performance was found comparable with the state-of-the-art algorithms [13].

The main goal of the study is to compare the performance of recently proposed variant of competitive differential evolution (CDE) [12] with the other well-performing adaptive DE algorithms on the hard benchmark test functions [9]. Another goal is to study the influence of small changes in competitive mechanism on the performance.

2 Differential Evolution

DE works with two alternating generations of population, P and Q. The points of population are considered as candidates of solution. At the beginning, the generation P is

V. Snasel et al. (Eds.): SOCO Models in Industrial & Environmental Appl., AISC 188, pp. 133–142.
springerlink.com

initialized randomly in the search domain S, $S = \prod_{j=1}^{D}[a_j, b_j],\ a_j < b_j,\ j = 1, 2, \ldots, D$. A new point y (trial point) is produced by mutation and crossover operations for each point $x_i \in P$, $i \in \{1, 2, \ldots, NP\}$, where NP is the size of population. Assuming minimization, the point y is inserted into new generation Q if $f(y) \leq f(x_i)$, otherwise the point x_i enters into Q. After completing the new generation, Q becomes the old generation P and the whole process continues until the stopping condition is satisfied. The basic scheme of DE is shown in a pseudo-code in Algorithm 1.

Algorithm 1. Differential evolution

```
generate an initial population P = (x_1, x_2, ..., x_NP), x_i ∈ S distributed uniformly
while stopping condition not reached do
  for i = 1 to NP do
    generate a trial vector y
    if f(y) ≤ f(x_i) then
      insert y into new generation Q
    else
      insert x_i into new generation Q
    end if
  end for
  P := Q
end while
```

The trial vector y is generated by crossover of two parent vectors, the current (target) vector x_i and a mutant vector v. The mutant vector v is obtained by a kind of mutation. Many kinds of mutation have been proposed, see e.g. [2,5,6,8], we mention those used in algorithms compared in this study. Suppose that r_1, r_2, r_3, r_4, and r_5 are five mutually distinct points taken randomly from population P, not coinciding with the current x_i, $F > 0$ is an input control parameter, and $\text{rand}(0,1)$ is a random number uniformly distributed between 0 and 1. The mutant vector v can be generated as follows:

- rand/1
$$v = r_1 + F(r_2 - r_3), \tag{1}$$
- rand/2
$$v = r_1 + F(r_2 - r_3) + F(r_4 - r_5), \tag{2}$$
- best/2
$$v = x_{\text{best}} + F(r_1 - r_2) + F(r_3 - r_4), \tag{3}$$
where x_{best} is the point with the minimum function value in the current population.
- rand-to-best/2
$$v = r_1 + F(x_{\text{best}} - r_1) + F(r_2 - r_3) + F(r_4 - r_5), \tag{4}$$
- current-to-rand/1
$$y = x_i + \text{rand}(0,1) \times (r_1 - x_i) + F(r_2 - r_3)\,. \tag{5}$$
Note that the current-to-rand/1 mutation generates a trial point y directly, because (5) includes so called arithmetic crossover.

- randrl/1

$$v = r_1^x + F\,(r_2 - r_3)\,, \tag{6}$$

  where the point r_1^x is not chosen randomly like in rand/1, but tournament best among r_1, r_2, and r_3, i.e. $r_1^x = \arg\min_{i\in\{1,2,3\}} f(r_i)$, as proposed in [3].
- current-to-pbest/1 [16]

$$v = x_i + F\,(x_{\text{pbest}} - x_i) + F\,(r_1 - r_2)\,, \tag{7}$$

  where x_{pbest} is randomly chosen from $100\,p\,\%$ best individuals with input parameter $p \in (0,1]$, value of $p \in [0.05, 0.20]$ is recommended. The vector $r_1 \neq x_i$ is randomly selected from P, r_2 is randomly selected from the union $P \bigcup A$ of the current population P and the archive A.

The crossover operator constructs the trial vector y from current individual x_i and the mutant vector v. Two types of crossover were proposed by Storn and Price in [8]. Binomial crossover replaces the elements of vector x_i using the following rule

$$y_j = \begin{cases} v_j & \text{if} \quad U_j \leq CR \quad \text{or} \quad j = l \\ x_{ij} & \text{if} \quad U_j > CR \quad \text{and} \quad j \neq l\,, \end{cases} \tag{8}$$

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 a control parameter influencing the number of elements to be exchanged by crossover. Eq. (8) ensures that at least one element of x_i is changed, even if $CR = 0$.

In exponential crossover, the starting position of crossover is also chosen randomly from $1, \ldots, D$, but L consecutive elements (counted in circular manner) are taken from the mutant vector v. Probability of replacing the kth element in the sequence $1, 2, \ldots, L$, $L \leq D$, decreases exponentially with increasing k. L adjacent elements are changed in exponential variant, in binomial one the changed coordinates are dispersed randomly over the coordinates $1, 2, \ldots, D$. While the relation between the probability of mutation and the CR is linear in binomial crossover, in the exponential crossover this relation is nonlinear and the deviation from linearity enlarges with increasing dimension of problem. Probability of mutation (p_m) controls the number of exchanged elements in crossover, $p_m \times D$ is the expected number of mutant elements used in producing offsprings. Zaharie [14,15] derived the relation between p_m and CR for exponential crossover. Her result can be rewritten in the form of polynomial equation

$$CR^D - D\,p_m\,CR + D\,p_m - 1 = 0\,. \tag{9}$$

The value of CR for given value of $p_m \in (1/D, 1)$ can be evaluated as the root of the equation (9).

The combination of mutation and crossover is called DE strategy, usually abbreviated by DE/m/n/c, where m stands for a kind of mutation, n for the number of differences of randomly selected points in mutation, and c for the type of crossover.

3 Competitive Differential Evolution

Adaptive mechanism for DE algorithm based on the competition of different strategies or different settings of $[F, CR]$ was introduced in [10]. Let us have H strategies or different $[F, CR]$ settings in a pool. For simplicity, we speak on H strategies in the pool. Any of H strategies can be chosen for the generation of a new trial point y. A strategy is selected randomly with probability $q_h,\ h = 1, 2, \ldots, H$. At the start the values of probability are set uniformly, $q_h = 1/H$, and they are modified according to their success rates in the preceding steps of the search process. The hth strategy is considered successful if it generates such a trial vector y satisfying $f(y) \leq f(x_i)$. Probability q_h is evaluated as the relative frequency according to

$$q_h = \frac{n_h + n_0}{\sum_{j=1}^{H}(n_j + n_0)}\,, \tag{10}$$

where n_h is the current count of the hth strategy successes, and $n_0 > 0$ is an input parameter. The setting of $n_0 > 1$ prevents from a dramatic change in q_h by one random successful use of the hth strategy. To avoid degeneration of the search process, the current values of q_h are reset to their starting values if any probability q_h decreases below some given limit $\delta,\ \delta > 0$.

Several variants of competitive DE differing both in the pool of DE strategies and in the set of control-parameters values were tested [11]. A variant of competitive DE appeared well-performing and robust in different benchmark tests in [12]. In this variant, denoted *b6e6rl* hereafter, 12 strategies are in competition ($H = 12$), six of them with the binomial crossover and six ones with the exponential crossover. The randrl/1 mutation (6) is applied in all the strategies. Two different values of control parameter F ($F = 0.5$ and $F = 0.8$) are combined with three values of CR, which gives six different setting for each crossover. The binomial crossover uses the values of $CR \in \{0, 0.5, 1\}$. The values of CR for exponential crossover are evaluated as the roots of the equation (9), corresponding three values of probability p_m are set up equidistantly in the interval $(1/D, 1)$. The input parameters controlling competition are standardly set up to $n_0 = 2$ and $\delta = 1/(5 \times H)$.

4 Experiments and Results

The adaptive DE algorithms experimentally compared in this study including the basic features of their adaptive mechanism are briefly summarized in Table 1.

Two modifications of b6e6rl algorithm are tested. In the first one, hereafter called b6e6rl60, the parameter δ controlling the competition was set to conventional value, i.e. $\delta = 1/(5 * H) = 1/60$. In the second one, hereafter called b6e6rl480, the parameter δ is set to a much less value, $\delta = 1/(40 * H) = 1/480$, in order to find out how the preference of successful strategies for a longer period influences the performance of the algorithm.

These two modifications of b6e6rl algorithm were applied to twenty five test functions defined for CEC2005 competition [9]. Tests were carried out for the $D = 30$ dimension of the problems under the experimental conditions specified in [13], 25

Table 1. Adaptive DE variants in experimental comparison

Algorithm	Strategy #	Strategy type	Adaptive mechanism
jDE [1]	1	rand/1/bin	evolutionary self-adaptation of F and CR with respect to their success
JADE [16]	1	curr-to-pbest/1/bin	adaptation of F and CR with respect to their success in the current generation
SADE [7]	4	rand/1/bin rand/2/bin rand-to-best/2/bin curr-to-rand/1	competition of strategies, F random without adaptation, CR – median of successful values in last LP generations
EPSDE [4]	3	rand/1/bin best/2/bin curr-to-rand/1	competition of strategies, evolutionary selection of successful strategy and control parameters, mutation by random selection from the pool of strategies and the pools of F and CR fixed values
CoDE [13]	3	rand/1/bin rand/2/bin curr-to-rand/1	F and CR selected at random from three pairs of fixed values, tournament selection among strategies
b6e6rl [12]	2	randrl/1/bin randrl/1/exp	competition of strategies with assigned fixed values of F and CR, probability of strategy selection proportional to its success, resetting the values of probability if any of probability values is too small

independent runs for each benchmark function were carried out, each run was stopped if the number of function evaluations $FES = 3 \times 10^5$ was achieved.

A comparison of b6e6rl algorithm modifications with other algorithms is presented in Tables 2 and 3. The average and standard deviation of the function error values are given here. The function error value in a run is computed as $f(x_{min}) - f(x^*)$, where x_{min} is the best solution found by the algorithm in a run and x^* is the global minimum of the function. The results of the other algorithms are taken from [13]. The minimum average error values for each function are printed in bold, all the error values less than 1×10^{-10} are considered equal. The numbers of wins are given in the last rows of the tables.

With respect to the number of wins, the best algorithms are CoDE, JADE, and b6e6rl60 in this order, followed by the medium-performing algorithms (b6e6rl480, EPSDE, and jDE), SADE appeared the worst performing.

A summarized comparison of the algorithms on 25 benchmark functions is shown in Table 4. The rank of the algorithm with respect to average function error was assigned to each algorithm for each function. All the error values less than 1×10^{-10} are again considered equal and were assigned by their average rank. The algorithms are presented in the ascending order of their average rank. We can see that best performing algorithms

are CoDE and JADE while other algorithms differ only very little and there is almost no difference in the performance of b6e6rl modifications.

Similar summarized comparison on the subset of easier test functions (F1 to F14) is depicted in Table 5. The order of the algorithms differs a little from the comparison on the whole set of functions. CoDE and JADE are again best performing but b6e6rl480 modification is the third best algorithm followed by b6e6rl60 with almost equal average rank. The other algorithms have their average rank greater by more than one.

The modifications of b6e6rl algorithm were also compared in the second set of experiments in order to investigate the ability of the algorithm to find an acceptable solution of the problem. The experiments were carried out on the easier subset of test functions (F1–F14), 25 runs were performed for each function. The higher maximum number of *FES* was allowed but a run was stopped if the solution found in the run was near to the correct solution of the problem. The stopping condition was set as follows:

$$(f(x_{min}) - f(x^*)) < \varepsilon \quad OR \quad FES > 3 \times 10^6$$

Table 2. Results of b6e6rl60, b6e6rl480, JADE, and jDE, $D = 30$, $FES = 300000$

	b6e6rl60		b6e6rl480		JADE		jDE	
F	mean	std	mean	std	mean	std	mean	std
F1	**0.00E+00**	0.0E+00	**0.00E+00**	0.0E+00	**0.00E+00**	0.0E+00	**0.00E+00**	0.0E+00
F2	**1.18E-13**	6.3E-14	**2.98E-13**	1.5E-13	**1.07E-28**	1.0E-28	1.11E-06	2.0E-06
F3	9.09E+04	5.3E+04	3.79E+05	2.2E+05	**8.42E+03**	7.3E+03	1.98E+05	1.1E+05
F4	**1.11E-13**	7.2E-14	5.77E-07	2.0E-06	**1.73E-16**	5.4E-16	4.40E-02	1.3E-01
F5	5.44E+02	5.4E+02	1.14E+03	6.7E+02	**8.59E-08**	5.2E-07	5.11E+02	4.4E+02
F6	**3.41E-14**	2.8E-14	**3.41E-14**	2.8E-14	1.02E+01	3.0E+01	2.35E+01	2.5E+01
F7	**6.30E-03**	7.7E-03	7.09E-03	8.6E-03	8.07E-03	7.4E-03	1.18E-02	7.8E-03
F8	2.10E+01	5.7E-02	2.09E+01	7.1E-02	2.09E+01	1.7E-01	2.09E+01	4.9E-02
F9	**0.00E+00**	0.0E+00	**0.00E+00**	0.0E+00	**0.00E+00**	0.0E+00	**0.00E+00**	0.0E+00
F10	6.38E+01	1.0E+01	4.79E+01	9.0E+00	**2.41E+01**	4.6E+00	5.54E+01	8.5E+00
F11	2.66E+01	2.1E+00	2.75E+01	1.6E+00	2.53E+01	1.7E+00	2.79E+01	1.6E+00
F12	1.48E+04	6.3E+03	1.16E+04	4.6E+03	6.15E+03	4.8E+03	8.63E+03	8.3E+03
F13	1.42E+00	1.2E-01	**1.25E+00**	8.3E-02	1.49E+00	1.1E-01	1.66E+00	1.4E-01
F14	1.26E+01	2.3E-01	1.25E+01	3.0E-01	**1.23E+01**	3.1E-01	1.30E+01	2.0E-01
F15	3.64E+02	1.2E+02	3.88E+02	8.3E+01	3.51E+02	1.3E+02	3.77E+02	8.0E+01
F16	1.32E+02	1.0E+02	9.46E+01	7.0E+01	1.01E+02	1.2E+02	7.94E+01	3.0E+01
F17	1.61E+02	7.1E+01	1.15E+02	2.6E+01	1.47E+02	1.3E+02	1.37E+02	3.8E+01
F18	9.05E+02	1.2E+00	9.06E+02	1.5E+00	9.04E+02	1.0E+00	9.04E+02	1.1E+01
F19	9.06E+02	1.7E+00	9.06E+02	2.0E+00	9.04E+02	8.4E-01	9.04E+02	1.1E+00
F20	9.05E+02	1.0E+00	9.06E+02	2.8E+00	9.04E+02	8.5E-01	9.04E+02	1.1E+00
F21	**5.00E+02**	1.2E-13	**5.00E+02**	1.2E-13	**5.00E+02**	4.7E-13	**5.00E+02**	4.8E-13
F22	8.82E+02	1.9E+01	8.88E+02	2.9E+01	8.66E+02	1.9E+01	8.75E+02	1.9E+01
F23	**5.34E+02**	3.6E-04	**5.34E+02**	3.9E-04	5.50E+02	8.1E+01	**5.34E+02**	2.8E-04
F24	**2.00E+02**	6.0E-13	**2.00E+02**	6.0E-13	**2.00E+02**	2.9E-14	**2.00E+02**	2.9E-14
F25	**2.11E+02**	1.1E+00	2.12E+02	1.1E+00	**2.11E+02**	7.9E-01	**2.11E+02**	7.3E-01
#wins	10		8		11		6	

Table 3. Results of SADE, EPSDE, and CoDE, $D = 30$, $FES = 300000$

	SADE		EPSDE		CoDE	
F	mean	std	mean	std	mean	std
F1	**0.00E+00**	0.0E+00	**0.00E+00**	0.0E+00	**0.00E+00**	0.0E+00
F2	8.26E-06	1.7E-05	**4.23E-26**	4.1E-26	**1.69E-15**	4.0E-15
F3	4.27E+05	2.1E+05	8.74E+05	3.3E+06	1.05E+05	6.3E+04
F4	1.77E+02	2.7E+02	3.49E+02	2.2E+03	5.81E-03	1.4E-02
F5	3.25E+03	5.9E+02	1.40E+03	7.1E+02	3.31E+02	3.4E+02
F6	5.31E+01	3.3E+01	6.38E-01	1.5E+00	1.60E-01	7.9E-01
F7	1.57E-02	1.4E-02	1.77E-02	1.3E-02	7.46E-03	8.6E-03
F8	2.09E+01	5.0E-02	2.09E+01	5.8E-02	**2.01E+01**	1.4E-01
F9	2.39E-01	4.3E-01	3.98E-02	2.0E-01	**0.00E+00**	0.0E+00
F10	4.72E+01	1.0E+01	5.36E+01	3.0E+01	4.15E+01	1.2E+01
F11	1.65E+01	2.4E+00	3.56E+01	3.9E+00	**1.18E+01**	3.4E+00
F12	**3.02E+03**	2.3E+03	3.58E+04	7.1E+03	3.05E+03	3.8E+03
F13	3.94E+00	2.8E-01	1.94E+00	1.5E-01	1.57E+00	3.3E-01
F14	1.26E+01	2.8E-01	1.35E+01	2.1E-01	**1.23E+01**	4.8E-01
F15	3.76E+02	7.8E+01	**2.12E+02**	2.0E+01	3.88E+02	6.9E+01
F16	8.57E+01	6.9E+01	1.22E+02	9.2E+01	**7.37E+01**	5.1E+01
F17	7.83E+01	3.8E+01	1.69E+02	1.0E+02	**6.67E+01**	2.1E+01
F18	8.68E+02	6.2E+01	**8.20E+02**	3.4E+00	9.04E+02	1.0E+00
F19	8.74E+02	6.2E+01	**8.21E+02**	3.4E+00	9.04E+02	9.4E-01
F20	8.78E+02	6.0E+01	**8.22E+02**	4.2E+00	9.04E+02	9.0E-01
F21	5.52E+02	1.8E+02	8.33E+02	1.0E+02	**5.00E+02**	4.9E-13
F22	9.36E+02	1.8E+01	**5.07E+02**	7.3E+00	8.63E+02	2.4E+01
F23	**5.34E+02**	3.6E-03	8.58E+02	6.8E+01	**5.34E+02**	4.1E-04
F24	**2.00E+02**	6.2E-13	2.13E+02	1.5E+00	**2.00E+02**	2.9E-14
F25	2.14E+02	2.0E+00	2.13E+02	2.6E+00	**2.11E+02**	9.0E-01
#wins	4		7		12	

Table 4. Comparison of algorithms performance according to their average rank on 25 benchmark functions

Algorithm	Average rank
CoDE	2.74
JADE	3.12
jDE	4.12
b6e6rl60	4.24
b6e6rl480	4.36
SaDE	4.44
EPSDE	4.98

Table 5. Comparison of algorithms performance according to their average rank on functions F1 to F14

Algorithm	Average rank
CoDE	2.61
JADE	2.68
b6e6rl480	3.61
b6e6rl60	3.64
jDE	4.75
SaDE	5.04
EPSDE	5.68

Table 6. Required accuracy ε for the benchmark functions

Functions	ε
F1 – F5	1×10^{-6}
F6 – F14	1×10^{-2}

Table 7. Reliability and average *FES* in successful runs for b6e6rl modifications on the easier subset of test functions

	b6e6rl60		b6e6rl480	
Function	R	Av. FES	R	Av. FES
F1	100	48682	100	44174
F2	100	69984	100	87403
F4	100	129533	100	229982
F6	100	140374	100	163003
F7	84	57483	72	59480
F9	100	53794	100	45209
F12	12	862940	4	1835400

where x_{min} is the best solution found in a run, x^* is the global minimum of the function, ε is the required accuracy prescribed in [9], and *FES* is the current number of function evolutions. The values of ε used in the experiments are given in Table 6. The results of this comparison obtained on the F1–F14 subset of test functions are shown in Table 7, the reliability of the search is given in the column *R*, which is the percentage of the runs that found a solution satisfying the condition $(f(x_{min}) - f(x^*)) < \varepsilon$. Only functions with $R > 0$ are presented in the table, in the other test problems no acceptable solution was found in 3×10^6 function evaluations. The results show if the b6e6rl algorithm can find an acceptable solution, it is able to find it faster than in 3×10^5 *FES* in most test problems. However, in 7 out of 14 problems the algorithm is not able to find any good solution even in 3×10^6 *FES*.

It is seen from the table that the preference of successful strategies for a longer period in b6e6rl480 does not bring better efficiency of the search. The average *FES* of b6e6rl480 is higher in 5 out of 7 functions. As it was expected, the preference of successful strategies for a longer period decreased the reliability (in 2 out of 7 functions).

5 Conclusion

The performance of b6e6rl variant of competitive DE appeared to be comparable with the state-of-the-art adaptive versions of differential evolution algorithm when applied to the hard benchmark problems [9]. With respect the number of wins, it was outperformed by CoDE and JADE only. The average ranks of both b6e6rl modifications are in the middle of the algorithms in the experimental comparison. In the subset of easier test functions, the performance of b6e6rl modifications is even better in their average ranks, outperformed only by CoDE and JADE.

The preference of successful strategies for a longer period by setting the control parameter of competition to $\delta = 1/480$ does not bring better efficiency of the search compared to its former recommended value $\delta = 1/60$. The recommended values of the parameters controlling the competitive adaptation should be used when considering the application of b6e6rl algorithm.

Acknowledgement. This work was supported by the European Regional Development Fund in the IT4Innovations Centre of Excellence project (CZ.1.05/1.1.00/02.0070) and partly supported by University of Ostrava, the project SGS13/PřF/2012.

References

1. Brest, J., Greiner, S., Boškovič, B., Mernik, M., Žumer, V.: Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. IEEE Transactions on Evolutionary Computation 10, 646–657 (2006)
2. Das, S., Suganthan, P.N.: Differential evolution: A survey of the state-of-the-art. IEEE Transactions on Evolutionary Computation 15, 27–54 (2011)
3. Kaelo, P., Ali, M.M.: A numerical study of some modified differential evolution algorithms. European J. Operational Research 169, 1176–1184 (2006)
4. Mallipeddi, R., Suganthan, P.N., Pan, Q.K., Tasgetiren, M.F.: Differential evolution algorithm with ensemble of parameters and mutation strategies. Applied Soft Computing 11, 1679–1696 (2011)
5. Neri, F., Tirronen, V.: Recent advances in differential evolution: a survey and experimental analysis. Artificial Intelligence Review 33, 61–106 (2010)
6. Price, K.V., Storn, R., Lampinen, J.: Differential Evolution: A Practical Approach to Global Optimization. Springer (2005)
7. Qin, A.K., Huang, V.L., Suganthan, P.N.: Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Transactions on Evolutionary Computation 13, 398–417 (2009)
8. Storn, R., Price, K.V.: Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. J. Global Optimization 11, 341–359 (1997)

9. Suganthan, P.N., Hansen, N., Liang, J.J., Deb, K., Chen, Y.P., Auger, A., Tiwari, S.: Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization (2005), http://www.ntu.edu.sg/home/epnsugan/
10. Tvrdík, J.: Competitive differential evolution. In: Matoušek, R., Ošmera, P. (eds.) MENDEL 2006, 12th International Conference on Soft Computing, pp. 7–12. University of Technology, Brno (2006)
11. Tvrdík, J.: Adaptation in differential evolution: A numerical comparison. Applied Soft Computing 9, 1149–1155 (2009)
12. Tvrdík, J.: Self-adaptive variants of differential evolution with exponential crossover. Analele of West University Timisoara, Series Mathematics-Informatics 47, 151–168 (2009), http://www1.osu.cz/~tvrdik/down/global_optimization.html
13. Wang, Y., Cai, Z., Zhang, Q.: Differential evolution with composite trial vector generation strategies and control parameters. IEEE Transactions on Evolutionary Computation 15, 55–66 (2011)
14. Zaharie, D.: A comparative analysis of crossover variants in differential evolution. In: Markowska-Kaczmar, U., Kwasnicka, H. (eds.) Proceedings of IMCSIT 2007, pp. 171–181. PTI, Wisla (2007)
15. Zaharie, D.: Influence of crossover on the behavior of differential evolution algorithms. Applied Soft Computing 9, 1126–1138 (2009)
16. Zhang, J., Sanderson, A.C.: JADE: Adaptive differential evolution with optional external archive. IEEE Transactions on Evolutionary Computation 13, 945–958 (2009)