Adaptive Variants of Differential Evolution: Towards Control-Parameter-Free Optimizers

Josef Tvrdík, Radka Poláková, Jiří Veselský, and Petr Bujok

Abstract. Seven up-to-date adaptive variants of differential evolution were compared in six benchmark problems of two levels of dimension ($D = 30$ and $D = 100$). The opposition-based optimization was also implemented to each adaptive variant and compared in experiments. It was found that all the algorithms perform very reliably in the problems of $D = 30$, whereas their reliability rate in the problems of $D = 100$ differs substantially among the test problems. Only two algorithms (JADE and *b6e6rl* variant of competitive DE) operate with acceptable reliability in all the problems. Considering the computational costs, the rank of the algorithms is different in various problems. When the average performance over all the problems is taken into account, JADE was the most efficient and *b6e6rl* the most reliable. The implementation of opposition-based optimization into adaptive variants of differential evolution does not increase the reliability and its positive influence on the efficiency is rare. Based on the results, recommendations to application of adaptive algorithms are formed and the source code of the algorithms is available online.

1 Introduction

The search of the global extreme of an objective function occurs frequently in many fields of human activities. Without loss of generality, the problem can be simply

Josef Tvrdík
Department of Computer Science and Centre of Excellence IT4Innovations,
Division of UO, University of Ostrava, Ostrava, Czech Republic
e-mail: `josef.tvrdik@osu.cz`

Radka Poláková
Department of Mathematics, University of Ostrava, Ostrava, Czech Republic
e-mail: `radka.polakova@wo.cz`

Jiří Veselský · Petr Bujok
Department of Computer Science, University of Ostrava, Ostrava, Czech Republic
e-mail: `jiri.veselsky@osu.cz`, `petr.bujok@osu.cz`

formed as minimization problem in a very clear form. The single-objective continuous optimization problem is formed as follows:

The objective function to be minimized is $f(\mathbf{x})$, $\mathbf{x} = (x_1, x_2, \ldots, x_D) \in \mathscr{R}^D$, the feasible domain Ω is defined by specifying boundary constraints, which are lower (a_j) and upper (b_j) limits of each component j, $\Omega = \prod_{j=1}^{D} [a_j, b_j]$, $a_j < b_j$, $j = 1, 2, \ldots, D$. The global minimum point $\mathbf{x}^*$ satisfying condition $f(\mathbf{x}^*) \leq f(\mathbf{x}), \forall \mathbf{x} \in \Omega$ is the solution of the problem. If needed, the global maximum of $f(\mathbf{x})$ can be found as the global minimum of $g(\mathbf{x}) = -f(\mathbf{x})$.

The problem is formulated clearly and the need to solve such a problem occurs frequently in many areas. Natural demand is to find an acceptable approximation of the global minimum point reliably and as quickly as possible. However, finding the point $\mathbf{x}^*$ is not an easy task. There is no deterministic algorithm solving this problem in polynomial time [2] in general. Standard deterministic optimization 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 the evolution in populations and they are called evolutionary algorithms (EAs). Such algorithms are able to find an acceptable solution sufficiently close to the global minimum point $\mathbf{x}^*$ with reasonable computational costs, but efficiency of the search is sensitive to the setting of their control parameters. Application of them usually requires a time-consuming tuning of control parameters to the problem in question.

Researchers have spent a great effort to develop adaptive or self-adaptive evolutionary algorithms applicable without control-parameter tuning, i.e. such algorithms that are almost control-parameter-free. In spite of the fact following from No Free Lunch Theorem [28] (no search algorithm is superior to others for all possible optimization problems) some self-adaptive algorithms performing well for a wide class of problems were developed.

2 Differential Evolution Algorithm

Differential evolution (DE) was introduced by Storn and Price [21, 22] as a global optimizer for continuous optimization problems with a real-value objective function. DE algorithm has become one of the most frequently evolutionary algorithms used for solving the global optimization problems in recent years [14]. Like other evolutionary algorithms, DE works with a population of individuals (*NP* points in the feasible domain Ω), that are considered as candidates of solution. The population is developing iteratively during the process by using evolutionary operators of selection, mutation, and crossover. The basic scheme of DE is shown in a pseudo-code in Algorithm 1.

The trial vector $\mathbf{y}$ is generated by crossover of two parent vectors, the current (target) vector $\mathbf{x}_i$ and a mutant vector $\mathbf{v}$. The mutant vector $\mathbf{v}$ is obtained by a mutation. During last years many kinds of mutation have been proposed and tested. Here we mention those kinds of mutation used in algorithms compared in this study. Suppose that $\mathbf{r}_1$, $\mathbf{r}_2$, $\mathbf{r}_3$, $\mathbf{r}_4$, and $\mathbf{r}_5$ are five mutually distinct points taken randomly from

Algorithm 1. Differential evolution

```
1: generate an initial population $P = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{NP})$, $\mathbf{x}_i \in \Omega$ distributed uniformly
2: while stopping condition not reached do
3:     for $i = 1$ to $NP$ do
4:         generate a trial vector $\mathbf{y}$
5:         if $f(\mathbf{y}) \le f(\mathbf{x}_i)$ then
6:             insert $\mathbf{y}$ into new generation $Q$
7:         else
8:             insert $\mathbf{x}_i$ into new generation $Q$
9:         end if
10:    end for
11:    $P := Q$
12: end while
```

population P, not coinciding with the current $\mathbf{x}_i$, $F > 0$ is a control parameter, and $\mathrm{rand}(0,1)$ is an uniformly distributed random number between 0 and 1. The mutation vector $\mathbf{v}$ can be generated as follows:

- rand/1
$$\mathbf{v} = \mathbf{r}_1 + F\left(\mathbf{r}_2 - \mathbf{r}_3\right), \tag{1}$$
- rand/2
$$\mathbf{v} = \mathbf{r}_1 + F\left(\mathbf{r}_2 - \mathbf{r}_3\right) + F\left(\mathbf{r}_4 - \mathbf{r}_5\right), \tag{2}$$
- best/2
$$\mathbf{v} = \mathbf{x}_{\text{best}} + F\left(\mathbf{r}_1 - \mathbf{r}_2\right) + F\left(\mathbf{r}_3 - \mathbf{r}_4\right), \tag{3}$$
  where $\mathbf{x}_{\text{best}}$ is the point with the minimum function value in the current population.
- rand-to-best/2
$$\mathbf{v} = \mathbf{r}_1 + F\left(\mathbf{x}_{\text{best}} - \mathbf{r}_1\right) + F\left(\mathbf{r}_2 - \mathbf{r}_3\right) + F\left(\mathbf{r}_4 - \mathbf{r}_5\right), \tag{4}$$
- current-to-rand/1
$$\mathbf{y} = \mathbf{x}_i + \mathrm{rand}(0,1) \times \left(\mathbf{r}_1 - \mathbf{x}_i\right) + F\left(\mathbf{r}_2 - \mathbf{r}_3\right). \tag{5}$$
  Note that the current-to-rand/1 mutation generates a trial point $\mathbf{y}$ directly, because (5) includes so called arithmetic crossover.
- randrl/1
$$\mathbf{v} = \mathbf{r}_1^x + F\left(\mathbf{r}_2 - \mathbf{r}_3\right), \tag{6}$$
  where the point $\mathbf{r}_1^x$ is not chosen randomly like in rand/1, but tournament best among $\mathbf{r}_1$, $\mathbf{r}_2$, and $\mathbf{r}_3$, i.e. $\mathbf{r}_1^x = \arg\min_{i \in \{1,2,3\}} f(\mathbf{r}_i)$, as proposed in [8].

The crossover operator constructs the trial vector $\mathbf{y}$ from current individual $\mathbf{x}_i$ and the mutant vector $\mathbf{v}$. Two types of crossover were proposed by founders of DE in [22]. Binomial crossover replaces the elements of vector $\mathbf{x}_i$ using the following rule

$$y_j = \begin{cases} v_j & \text{if} \quad U_j \le CR \quad \text{or} \quad j = l \\ x_{ij} & \text{if} \quad U_j > CR \quad \text{and} \quad j \ne l, \end{cases} \tag{7}$$

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. (7) ensures that at least one element of $\mathbf{x}_i$ is changed, even if $CR = 0$. The variant of DE using mutation (1) and binomial crossover, in abbreviation DE/rand/1/bin, is the most frequently used DE strategy in applications.

For exponential crossover (strategies with the exponential crossover are denoted by 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 $\mathbf{v}$. Probability of replacing the kth element in the sequence $1, 2, \ldots, L,\ \ L \le D$, 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 in binomial crossover the relation between the probability of mutation and the CR is linear, 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 mean value of mutant elements' count used in producing offsprings. Zaharie [29, 30] derived the relation between p_m and CR for exponential crossover. Her result can be rewritten in the form of polynomial equation

$$CR^D - D\, p_m\, CR + D\, p_m - 1 = 0\,. \tag{8}$$

The value of CR for given value of $p_m \in (1/D, 1)$ can be evaluated as the root of the equation (8). The exponential crossover resembles two-point crossover in genetic algorithms but the length of exchanged part is controlled by the parameter CR.

Compared to other evolutionary algorithms (EAs), the differential evolution has a very few control parameters. Except the size of population NP common for all EAs it is the choice of mutation and crossover strategy, and pair of parameters F and CR, controlling the mutation and crossover, respectively. However, the efficiency of differential evolution is very sensitive to the control parameter setting of F and CR values and partly also to the selection of a DE strategy. Suitable control-parameter values for a specific problem may be found by trial-and-error tuning, but it requires a lot of time.

Based on wide experimental results, there are some recommendations for the setting of these parameters, see e.g. [6, 7, 14, 19, 22], but such recommendations are not valid in general.

In order to avoid time-consuming parameter tuning in the applications of DE, several new adaptive or self-adaptive modifications of DE were proposed, e.g. [3, 4, 9, 10, 12, 15, 16, 24, 25, 31, 32]. Moreover, the application of opposition-based learning in DE is proposed in [17, 18] and it was found that it increases the performance of non-adaptive DE variants. The up-to-date summary of the results in DE research has been presented recently in the comprehensive papers by Das and Suganthan [5] and by Neri and Tirronen [11].

3 DE Variants in Experimental Comparison

Four self-adaptive DE variants (*jDE* [3], *JADE* [32], *SaDE* [15], and *EPSDE* [10]) are considered currently as the state-of-the-art DE variants and the performance of novel DE variants is compared with these state-of-the-art DE variants in currently appearing studies. These four DE variants are also included in our study along with a variant of competitive DE [26] and two variants of DE based on composite trial vector generation strategies and control parameters that has been published recently [27].

3.1 jDE

A simple and efficient adaptive DE variant (mostly called "jDE" in literature) was proposed by Brest et al. [3]. It uses the DE/rand/1/bin with an evolutionary self-adaptation of F and CR. The tuple of these control parameters is encoded with each individual of the population and survives if an individual is successful, i.e. if it generates such a trial vector which is inserted into next generation.

The values of F and CR are initialized randomly for each point in population and survive with the individuals in the population, but they can be randomly mutated in each generation with given probabilities τ_1 and τ_2. If the mutation condition happens, new values of $CR \in [0, 1]$ uniformly distributed, and F also distributed uniformly in $[F_l, F_u]$ are used in generating a trial vector and stored in the new population. Input parameters are set to $F_l = 0.1$, $F_u = 0.9$, $\tau_1 = 0.1$, and $\tau_2 = 0.1$ as applied in [3].

3.2 SaDE

Differential evolution algorithm with strategy adaptation (SaDE) was introduced by Qin and Suganthan in [16]. A more sophisticated and more efficient variant was proposed later in [15] and it is used in our experimental comparison.

Four mutation strategies, namely rand/1/bin, rand/2/bin, rand-to-best/2/bin, and current-to-rand/1, for creating new trial vectors are stored in a strategy pool. Initially the probabilities of all the strategies are set to $1/4$, i.e. all the strategies have the same probability to be chosen. After the first LP generations, the probability of strategy selection to generate a new trial vector is based on its success rate in previous LP generations, the LP generations are used as a learning period. The calculation of probability values is carried out as follows:

$$p_k = \frac{S_k}{\sum_{j=1}^{4} S_j}\,, \tag{9}$$

where

$$S_k = \frac{succ_k}{succ_k + fail_k} + \varphi\,, \tag{10}$$

$succ_k$ is the cumulative count of the kth strategy success in previous LP generations. Similarly, $fail_k$ is the cumulative count of the kth strategy failure in generating a trial vector during last LP generations. Note that $succ_k + fail_k$ is the count of kth strategy selection during the learning period. The value of $\varphi = 0.01$ is used to avoid the possible null success rates. The values of p_k are recalculated after each generation.

The strategy selection in SaDE is similar to the selection applied in competitive DE as described in Subsection 3.5 but the evaluation of probability values is slightly different as well as the length of the learning period which here is a constant given as input parameter of the SaDE algorithm (set to recommended value $LP = 50$ in our experiments).

The values of the parameter F are generated randomly for each trial vector from a normal distribution with mean 0.5 and standard deviation 0.3, no adaptation of F is used in this algorithm.

The values of the parameter CR are generated from the normal distributions $N(CRm_k, 0.1)$, where the parameters CRm_k, $k = 1,2,3,4$, are the mean values of the distributions, the standard deviation is equal 0.1 for all the strategies. Initial values of $CRm_k = 0.5$ are used for all the strategies during the first LP generations and applied to those target vectors to which the kth strategy is assigned. To adapt the crossover rate CR, $CRmem_k$ vectors of length LP are used to store those CR values generating trial vectors successfully entering the next generation within previous LP generations. Then the values of CRm_k, $k = 1,2,3,4$, are updated to be the median of $CRmem_k$ after each generation.

3.3 JADE

JADE, introduced by Zhang and Sanderson in [32], is a new algorithm of adaptive differential evolution. It extends the original DE concept with three different improvements - current-to-pbest mutation strategy, adaptive control of parameters F and CR, and archive.

Current-to-best is one of the well-known mutation strategy used in various DE algorithms. Greedy strategies like current-to-best are known for their fast convergence as the best solution found so far is used in the evolutionary search. Due to the resultant reduction of the population density, these strategies may also cause problems such as premature convergence. In JADE, current-to-pbest mutation strategy with optional archive is used to provide fast convergence without losing reliability. The current mutant vector $\mathbf{v}$ is generated in the following manner:

$$\mathbf{v} = \mathbf{x}_i + F\left(\mathbf{x}_{\text{pbest}} - \mathbf{x}_i\right) + F\left(\mathbf{x}_{r_1} - \mathbf{x}_{r_2}\right), \tag{11}$$

where $\mathbf{x}_{\text{pbest}}$ is randomly chosen from $100p$ % best individuals with input parameter $p \in (0,1]$. Value of $p \in [0.05, 0.20]$ is recommended in [32], in our simulations we used $p = 0.05$. The vector $\mathbf{x}_{r_1}$ is randomly selected from P ($r_1 \neq i$), $\mathbf{x}_{r_2}$ is randomly selected from the union $P \bigcup A$ of the current population P and the archive A. The archive A is initialized as an empty set. In every generation, parent individuals replaced by better offspring individuals are put into the archive. After every generation the archive size is reduced to NP individuals by randomly dropping surplus individuals. The trial vector $\mathbf{y}$ is generated from $\mathbf{v}$ and $\mathbf{x}_i$ using the binomial crossover with the control parameter CR.

Adaptation of parameters CR and F is carried out as follows. For each generation, the crossover probability CR and the mutation factor F are independently generated for each individual $\mathbf{x}_i$. The CR value is generated according to the normal distribution of mean μ_{CR} and standard deviation 0.1 and then truncated to $[0,1]$. The F value is generated according to the Cauchy distribution with location factor μ_F and scale parameter 0.1. Then it is truncated to 1 if $F > 1$ or regenerated if $F < 0$. During every generation we keep S_{CR} as the set of all successful CR's in the generation and S_F as the set of all successful F's in the generation. The μ_{CR} and μ_F parameters are initialized to be 0.5 and updated at the end of each generation as

$$\begin{aligned} \mu_{CR} &\leftarrow (1-c)\,\mu_{CR} + c\,\text{mean}_A(S_{CR})\,, \\ \mu_F &\leftarrow (1-c)\,\mu_F + c\,\text{mean}_L(S_F)\,, \end{aligned} \tag{12}$$

where c is an algorithm parameter, $c \in [0,1]$ (we used $c = 0.1$), mean_A is the arithmetic mean, and mean_L is the Lehmer mean, defined as

$$\text{mean}_L(S_F) = \frac{\sum_{F \in S_F} F^2}{\sum_{F \in S_F} F}\,. \tag{13}$$

3.4 EPSDE

In this adaptive DE variant [10], an ensemble of mutation strategies and parameter values is applied. The mutation strategies and the values of control parameters are chosen from pools. The combination of the strategies and the parameters in the pools should have diverse characteristics, so that they can exhibit distinct performance during different stages of evolution when dealing with a particular problem. The triplet of *strategy*, F, CR is encoded along with each individual (vector) of population. At the beginning, the triplets are initialized randomly and then they develop by evolution. If the target vector produces a successful trial vector entering the next generation, its triplet (*strategy*, F, CR) survives with the trial vector for next generation and the successful triplet is also stored in auxiliary memory. Otherwise, the triplet (*strategy*, F, CR) is randomly re-initialized with a new mutation strategy and associated parameters from the respective pools or from the stored successful triplets with equal probability.

The following ensemble of strategies and values of control parameters are used in the EPSDE algorithm described in [10]:

- pool of strategies is {*best/2/bin, rand/1/bin, current-to-rand/1*},
- pool of F values is $\{0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$,
- pool of CR values is $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$.

The memory of the successful triplets has the length of LP, where LP is an input parameter of the algorithm set up to $LP = NP$ in our experiments.

Thus, the adaptive features of EPSDE can be considered as a combination of multiple strategies used in SaDE [15], competitive DE (e.g. *b6e6rl*) [26], and CoDE [27] with evolutionary approach to surviving the strategies successful in previous generation applied in jDE [3]. The advantage of EPSDE algorithm is its clear motivation and simplicity.

3.5 Competitive DE

Adaptive DE with competition of different strategies (*competitive DE*) was introduced in [24]. Any of H strategies in the pool can be chosen for the generation of a new trial point $\mathbf{y}$. 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 setting is considered successful if it generates such a trial vector $\mathbf{y}$ satisfying $f(\mathbf{y}) \le f(\mathbf{x}_i)$. Probability q_h is evaluated as the relative frequency according to

$$q_h = \frac{n_h + n_0}{\sum_{j=1}^{H}(n_j + n_0)}\,, \tag{14}$$

where n_h is the current count of the hth setting 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 $\delta,\ \delta > 0$. The input parameters controlling competition are recommended to be set up to $n_0 = 2$ and $\delta = 1/(5 \times H)$. These values are also used in our study.

Several variants of competitive DE differing both in the pool of DE strategies and in the set of control-parameters values were tested [25]. We use a variant of competitive DE that appeared well-performing and robust in different benchmark tests [26]. In this variant, denoted *b6e6rl* hereafter, 12 strategies are in competition ($H = 12$), six of them using the binomial crossover, rest of them using the exponential crossover. The randrl/1 mutation (6) is applied in all the strategies, two different values of control parameter F are used, $F = 0.5$ and $F = 0.8$. The binomial crossover uses three different values of *CR*, $CR \in \{0, 0.5, 1\}$. The values of *CR* for exponential crossover are evaluated as the roots of the equation (8), three values of probability p_m are set up equidistantly in the interval $(1/D, 1)$. These 12 very different competing strategies (small and medium value of F, six very different crossover operators) give a good chance for balancing exploration and exploitation in the different stages of the search process.

3.6 Composite Trial Vector Generation Strategies and Control Parameters

Another algorithm used in the experiments is DE with composite trial vector generation strategies and control parameters, CoDE, presented recently by Wang et al. [27], where CoDE algorithm was also compared with four adaptive DE variants (jDE, SaDE, JADE, EPSDE) in the extensive benchmark tests [23]. The results showed that CoDE is at least competitive with the algorithms in the comparison.

The CoDE combines three well-studied trial vector strategies with three control parameter settings in a random way to generate trial vectors. The strategies are rand/1/bin, rand/2/bin, and current-to-rand/1 and all the three strategies are applied when generating a new vector (line 4 in Algorithm 1). It results in having three offspring vectors and among them the vector with the least function value is used as the trial vector. The values of control parameters F and CR are chosen randomly from the parameter pool containing $[F = 1.0,\ CR = 0.1]$, $[F = 1.0,\ CR = 0.9]$, and $[F = 0.8,\ CR = 0.2]$.

After the first reading of [27] it was not quite clear if the binomial crossover should be used after the current-to-rand/1 mutation like in [13] or no other crossover is applied when this mutation is used because it includes so called arithmetic crossover making this strategy rotation invariant. Two versions of this algorithm were implemented, one with the binomial crossover (denoted "CoDE0" hereafter) and the second one without the binomial crossover (denoted "CoDE1" hereafter). The CoDE0 variant has appeared to be more efficient in preliminary experiments. In electronic discussion with the first author of the paper [27] it was found that the CoDE1 variant with small modifications was used in their experiments.

3.7 Opposition-Based DE

Application of opposition-based learning in stochastic optimization algorithms has appeared recently [17, 18] and it is called opposition-based optimization (OBO). Basic idea is to search for a solution not only within the individuals of the population developed by evolutionary operators but also in the opposite part of the current search space T. The opposite point to the point $\mathbf{x} \in T$, $T = \prod_{j=1}^{D} [l_j, u_j]$, $l_j < u_j$, $j = 1, 2, \ldots, D$ is defined as symmetric with respect to the center of T by

$$\breve{\mathbf{x}} = \mathbf{l} + \mathbf{u} - \mathbf{x}. \tag{15}$$

The opposite population O of NP points to the population P according to relation (15) is generated occasionally, then the function values in the opposite points are evaluated and from the set $P \cup O$ the NP fittest individuals are selected to the new population.

At the start of the optimization, the space for searching an opposite population is set to $T = \Omega = \prod_{j=1}^{D} [a_j, b_j]$, $a_j < b_j$, $j = 1, 2, \ldots, D$, i.e. the lower and upper boundaries are $\mathbf{l} = \mathbf{a}$ and $\mathbf{u} = \mathbf{b}$, respectively. The search space T for the opposite points shrinks dynamically during the search process. If the current population P

is a matrix of the size $(NP \times D)$, then lower and upper boundaries are evaluated as $\mathbf{l} = \min(P)$ and $\mathbf{u} = \max(P)$ supposing the functions $\min(\cdot)$ and $\max(\cdot)$ give the vectors of column minima and maxima, respectively. Thus, the current dynamically shrinking search space for the opposite population can be written as

$$T = \prod_{j=1}^{D} [l_j, u_j], \quad j = 1, 2, \ldots, D\,. \tag{16}$$

The opposite population is generated after the initialization of the population for the first time. During the search process, the opposite population is generated in randomly selected generations if the condition of $\text{rand}(0,1) < JR$ is satisfied, where $\text{rand}(0,1)$ is a random value from uniform distribution on $[0,1)$ and JR (jumping rate) is an input parameter of the algorithm. According to the results of experiments in [17], the jumping rate should be usually set to a constant value in the range $[0.1, 0.4]$ and rather small values of the jumping rates are recommended for smaller population sizes. The value of $JR = 0.3$ is used in our experiments.

The opposition-based differential evolution (ODE) was studied in [17]. The strategies DE/rand/1/bin, DE/rand/1/exp, DE/rand/2/bin, DE/rand/2/exp were experimentally compared in variants with and without OBO. It was found that ODE variants performed the same or better than their DE counterparts in respect to the number of objective function evaluations and the average reliability rate. It was also found that ODE outperformed the DE variant, where the random points in the dynamically shrinking search space T were used instead of the opposite points.

To our best knowledge there is no evidence if the opposition-based optimization might be helpful in adaptive DE variants. Hence the opposition-based optimization was implemented to all the adaptive DE variants described above in Subsections 3.1-3.6 and the performance of the corresponding algorithms with and without OBO was also compared.

4 Benchmark Functions

Six well-known test functions [1, 14, 22] are used as benchmark for all the DE variants in comparison. Four of the following test functions in their original non-shifted form have the global minimum point in the center of domain Ω, $\mathbf{x}^* = (0, 0, \ldots, 0)$, which makes the search of the solution easier for many stochastic algorithms. In this study, they were used in their shifted versions. The shifted function is evaluated at the point $\mathbf{z} = \mathbf{x} - \mathbf{o}$, $\mathbf{o} \in \Omega$, $\mathbf{o} \neq (0, 0, \ldots, 0)$. The shift $\mathbf{o}$ is generated randomly from the uniform D-dimensional distribution before each run.

- Shifted Ackley function - multimodal, separable:

$$f(\mathbf{z}) = -20 \exp\left(-0.02\sqrt{\tfrac{1}{D}\textstyle\sum_{j=1}^{D}(x_j - o_j)^2}\right) - \exp\left(\tfrac{1}{D}\textstyle\sum_{j=1}^{D}\cos 2\pi(x_j - o_j)\right) + \\ +20 + \exp(1)\,,$$

$x_j \in [-30, 30]$, $f(\mathbf{z}^*) = 0$, $\mathbf{x}^* = \mathbf{o}$.

- Shifted first DeJong function (sphere model) - unimodal, separable, convex, easy problem:

$$f(\mathbf{z}) = \sum_{j=1}^{D} (x_j - o_j)^2,$$

$x_j \in [-5.12, 5.12]$, $f(\mathbf{z}^*) = 0$, $\mathbf{x}^* = \mathbf{o}$.

- Shifted Griewank function - multimodal, nonseparable:

$$f(\mathbf{z}) = \sum_{j=1}^{D} \frac{(x_j - o_j)^2}{4000} - \prod_{j=1}^{D} \cos\left(\frac{(x_j - o_j)}{\sqrt{j}}\right) + 1,$$

$x_j \in [-400, 400]$, $f(\mathbf{z}^*) = 0$, $\mathbf{x}^* = \mathbf{o}$.

- Shifted Rastrigin function - multimodal, separable:

$$f(\mathbf{z}) = 10D + \sum_{j=1}^{D} \left[(x_j - o_j)^2 - 10\cos(2\pi(x_j - o_j))\right],$$

$x_j \in [-5.12, 5.12]$, $f(\mathbf{z}^*) = 0$, $\mathbf{x}^* = \mathbf{o}$.

- Rosenbrock function (second DeJong function, banana valley) - multimodal for $D > 3$ [20], nonseparable:

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

$x_j \in [-2.048, 2.048]$, $f(\mathbf{x}^*) = 0$, $\mathbf{x}^* = (1, 1, \ldots, 1)$.

- Schwefel function - multimodal, separable, the global minimum is distant from the next best local minima:

$$f(\mathbf{x}) = 418.98288727\, D - \sum_{j=1}^{D} x_j \sin(\sqrt{|\, x_j\,|}),$$

$x_j \in [-500, 500]$, $f(\mathbf{x}^*) \doteq 0$, $\mathbf{x}^* = (s, s, \ldots, s)$, $s \doteq 420.968746$.

5 Experiments

Seven adaptive DE variants described above and their counterparts with the extension of opposition-based optimization were experimentally compared in six benchmark problems of two levels of dimension, $D = 30$ and $D = 100$. The DE variants used in the experimental comparison are summarized in the following list:

jDE [3]	One DE strategy (DE/rand/1/bin) with self-adaptation of F and CR. Other control parameters are set to the values recommended by authors of the algorithm, i.e. the ranges of F and CR values are [0.1, 0.9] and [0, 1], respectively; mutation probabilities of F and CR are set to $\tau_1 = \tau_2 = 0.1$.

b6e6rl [26]	Twelve strategies differing in the type of crossover or the values of F and CR from the strategy pool compete to be selected for the generation of a new trial vector. The probability of a strategy selection is proportional to the previous performance of the strategy. The additional parameters controlling the competition are set to their recommended values, $n_0 = 2$ and $\delta = 1/60$.
SaDE [15]	Mutation strategy and the parameter CR are self-adapted based on their previous performance during last LP generations, F is generated randomly from the normal distribution with a mean and standard deviation of 0.5 and 0.3, respectively. The value of $LP = 50$ is used in our experiments.
JADE [32]	DE variant using a newly proposed current-to-pbest mutation with external archive and self-adaptation of the values of F and CR. The size of the archive is set to NP, $\mathbf{x}_{\text{pbest}}$ is randomly chosen from $100p\%$ best individuals, the value of $p = 0.05$ is used in our experiments, the $c = 0.1$ is used for the F and CR adaptation.
EPSDE [10]	Self-adaptive DE using an ensemble of mutation strategies and parameter values. The triplet $(\textit{strategy}, F, CR)$ is encoded along with each individual of the population. If the target vector produces a successful trial vector entering the next generation, its triplet $(\textit{strategy}, F, CR)$ survives to next generation and the successful triplet is also stored in auxiliary memory. Otherwise, the triplet $(\textit{strategy}, F, CR)$ is randomly re-initialized from the respective pools or from the stored successful triplets. The length of memory with the successful triplets is set to $LP = NP$.
CoDE1	Composite DE variant described in the paper by Wang et al. [27]. Three mutation strategies with parameters assigned randomly from respective pools are used in each attempt to generate an individual for next generation. The best point of the triple is used as a new trial point. No other adaptation is used.
CoDE0	Composite DE variant [27]. It differs from CoDE1 by the application of the binomial crossover after the current-to-rand/1 mutation.

Each DE variant in this list has also its counterpart with opposition-based optimization using the jumping rate set to $JR = 0.3$.

The DE variants could be considered almost control-parameter free because the auxiliary control parameters specific to each variant are set to their recommended values and used in all the problems. Thus, the size of the population and the stopping condition are the only control parameters to be set up. The same size of population was used for each DE variant, $NP = 60$. The stopping condition was defined the same for all the DE variants in tests as well. In contrast to the stopping condition of the maximum allowed number of function evaluations frequently used in literature, we formed a more realistic condition corresponding to practical optimization problems when we have no a priori knowledge on the appropriate computational costs. In such cases, the search should be finished before reaching the maximum allowed function evaluations if the difference in the worst and best individuals in population is small. It indicates that the continuation of the search can hardly find a significantly better solution. Thus, the stopping condition is defined in the form as it follows:

$$f_{\max} - f_{\min} < \varepsilon_f \quad \text{OR} \quad nfe > D \times maxevals\,, \tag{17}$$

where $f_{\max} - f_{\min}$ is the difference between the function values of the worst and the best individual in the population, *nfe* is the current number of function evaluations, ε_f and *maxevals* are input parameters set up to $\varepsilon_f = 1 \times 10^{-6}$ and $maxevals = 2 \times 10^4$, respectively, in the experiments.

One hundred of independent runs were carried out for each test problem and algorithm variant. The number of the function evaluations (*nfe*) and the minimum function value in the final generation ($f_{\min}$) were recorded in each run. The solution found in a run is considered acceptable if the minimum function value in the final generation does not differ from the known correct solution of the test problem by more than 1×10^{-4} (applicable only in test problems with known function value in the global minimum point). The reliability rate (R) of an algorithm in the solved problem was then evaluated as the number of runs when $f_{\min} - f(\mathbf{x}^*) < 1 \times 10^{-4}$, which means that R is the percentage of runs finding an acceptable solution. If the acceptable solution was found in a run, the number of the function evaluations needed for it was recorded. Moreover, the number of the successful trial vectors (i.e. satisfying the condition $f(\mathbf{y}) \le f(\mathbf{x}_i)$) was also recorded in each run.

6 Results

Results of the algorithms without using OBO are presented in the following subsection including their thorough analysis. The basic characteristics of performance of the algorithms using OBO are briefly presented in the next subsection together with the statistical evaluation of the influence of OBO on the algorithms performance. Selected additional characteristics of the search process are shown in the last subsection.

6.1 Performance of the Algorithms without OBO

Reliability rates (R) and mean values of the *nfe* of the tested DE variants without OBO are shown for the problems with $D = 30$ and $D = 100$ in Tables 1 and 3, respectively, the mean function values, the standard deviations, and minimum function values found in 100 runs are depicted in Tables 2 and 4.

It is obvious from the results with basic characteristics of computational costs and reliability of the search in Tables 1 and 3 that the performance of DE variants differs very substantially and the differences are problem-dependent. The CoDE1 is excluded from next considerations because of its bad overall performance. It was not able to find any acceptable solution before reaching the maximum *nfe* in half of the test problems and its computational costs exceed the other algorithms several times in the most of the problems. Hence it is hardly recommendable for applications solving the real-world problems.

Table 1 Basic characteristics of performance, $D = 30$

	Ackley		Dejong1		Griewank		Rastrigin		Rosenbrock		Schwefel	
	nfe	*R*	*nfe*	*R*	*nfe*	*R*	*nfe*	*R*	*nfe*	*R*	*nfe*	*R*
jDE	57592	100	32559	100	43690	100	137343	98	377216	97	60093	99
b6e6rl	71297	100	37472	100	51934	100	73402	100	147185	100	64243	100
SaDE	34594	75	20947	100	28312	87	79901	100	241504	95	53004	100
JADE	75248	100	13470	100	22759	93	67801	100	76440	93	57994	77
EPSDE	44899	100	23818	100	32438	100	251678	100	163082	100	74555	99
CoDE1	338816	100	177328	100	275249	100	410113	100	600000	0	300561	100
CoDE0	91426	100	47670	100	68010	100	268062	100	359860	100	149849	100

Table 2 Function values found by DE variants - means, standard deviations, and minima computed of 100 runs, $D = 30$

	Ackley			Dejong1			Griewank		
	mean	sd	min	mean	sd	min	mean	sd	min
jDE	1.21e-6	2.39e-7	5.13e-7	3.87e-7	1.03e-7	1.31e-7	3.99e-7	1.06e-7	1.71e-7
b6e6rl	1.67e-6	2.76e-7	1.01e-6	5.91e-7	1.36e-7	3.16e-7	5.93e-7	1.48e-7	2.08e-7
SaDE	1.04e-1	1.86e-1	7.90e-7	5.88e-7	1.52e-7	3.30e-7	1.67e-3	5.16e-3	1.14e-8
JADE	1.33e-6	2.30e-7	8.04e-7	4.78e-7	1.19e-7	2.40e-7	5.43e-4	2.00e-3	5.42e-20
EPSDE	1.33e-6	2.32e-7	7.97e-7	4.55e-7	1.06e-7	2.28e-7	4.53e-7	1.14e-7	2.61e-7
CoDE1	1.70e-6	2.73e-7	1.13e-6	6.31e-7	1.33e-7	3.37e-7	6.38e-7	1.17e-7	3.64e-7
CoDE0	1.26e-6	2.32e-7	6.50e-7	4.39e-7	1.03e-7	1.99e-7	4.36e-7	1.16e-7	2.10e-7
	Rastrigin			Rosenbrock			Schwefel		
jDE	1.99e-2	1.40e-1	1.84e-7	1.20e-1	6.83e-1	5.02e-8	1.18e+0	1.18e+1	1.69e-7
b6e6rl	5.84e-7	1.27e-7	2.83e-7	5.97e-7	1.66e-7	2.28e-7	5.94e-7	1.32e-7	3.25e-7
SaDE	4.80e-7	1.33e-7	1.79e-7	1.20e-1	6.83e-1	1.25e-6	4.68e-7	1.22e-7	2.45e-7
JADE	4.53e-7	1.34e-7	7.24e-8	2.79e-1	1.02e+0	3.80e-7	2.84e+1	5.36e+1	2.19e-7
EPSDE	4.49e-7	1.26e-7	1.32e-7	1.70e-6	8.58e-7	6.44e-7	1.18e+0	1.18e+1	2.10e-7
CoDE1	6.35e-7	1.23e-7	3.17e-7	1.21e+0	4.64e-1	4.96e-1	6.70e-7	1.33e-7	3.49e-7
CoDE0	4.09e-7	9.04e-8	2.23e-7	1.36e-6	4.49e-7	6.14e-7	4.47e-7	1.12e-7	2.34e-7

In the problems with $D = 30$, the remaining algorithms perform satisfactory with minimum reliability rate $R = 75\,\%$. They differ in convergence speed, JADE and

Table 3 Basic characteristics of performance, $D = 100$

	Ackley		Dejong1		Griewank		Rastrigin		Rosenbrock		Schwefel	
	nfe	*R*	*nfe*	*R*	*nfe*	*R*	*nfe*	*R*	*nfe*	*R*	*nfe*	*R*
jDE	151353	100	89183	100	109712	99	1071669	92	2000040	0	233726	87
b6e6rl	258244	100	145163	100	178750	99	271464	100	910790	97	248053	98
SaDE	77723	0	70955	100	87292	70	322024	91	1885897	51	179603	100
JADE	232678	100	32883	100	41293	65	219513	100	518355	69	174830	54
EPSDE	102604	89	61112	100	76005	82	2000040	0	1548091	82	612686	98
CoDE1	2000040	0	1993438	100	2000040	0	2000040	0	2000040	0	2000040	0
CoDE0	207726	97	123364	100	152927	88	2000040	0	2000040	0	1873010	99

Table 4 Function values found by DE variants - means, standard deviations, and minima computed of 100 runs, $D = 100$

	Ackley			Dejong1			Griewank		
	mean	sd	min	mean	sd	min	mean	sd	min
jDE	2.48e-6	3.45e-7	1.61e-6	9.63e-7	1.90e-7	6.59e-7	7.49e-5	7.40e-4	6.70e-7
b6e6rl	3.42e-6	5.21e-7	2.35e-6	1.31e-6	3.00e-7	5.90e-7	7.54e-5	7.40e-4	5.50e-7
SaDE	2.11e+0	4.84e-1	8.79e-1	3.48e-6	1.05e-6	1.78e-6	1.08e-2	3.02e-2	1.97e-6
JADE	4.45e-6	8.40e-7	3.02e-6	3.05e-6	8.72e-7	1.58e-6	7.46e-3	1.49e-2	1.79e-6
EPSDE	1.06e-1	3.05e-1	2.87e-6	2.01e-6	4.77e-7	1.23e-6	1.97e-3	4.46e-3	1.27e-6
CoDE1	3.85e-3	6.77e-4	2.56e-3	2.62e-6	5.78e-7	1.43e-6	2.98e-4	1.04e-4	1.45e-4
CoDE0	2.86e-2	1.64e-1	2.97e-6	1.98e-6	3.80e-7	1.21e-6	1.11e-3	3.14e-3	1.27e-6
	Rastrigin			Rosenbrock			Schwefel		
jDE	7.96e-2	2.71e-1	5.56e-7	2.02e+1	1.39e+1	6.82e+0	1.90e+1	5.51e+1	5.14e-7
b6e6rl	1.38e-6	2.62e-7	7.30e-7	1.20e-1	6.83e-1	7.93e-7	2.37e+0	1.67e+1	7.08e-7
SaDE	2.59e-1	1.47e+0	1.41e-6	1.11e+0	4.54e+0	2.35e-5	1.18e+0	1.18e+1	1.79e-6
JADE	1.77e-6	3.39e-7	1.07e-6	1.51e+0	9.91e+0	8.59e-6	6.28e+1	7.61e+1	1.23e-6
EPSDE	2.75e+2	1.38e+1	2.45e+2	1.99e-1	8.73e-1	5.88e-6	2.37e+0	1.67e+1	8.75e-7
CoDE1	1.82e+2	1.13e+1	1.59e+2	6.24e+2	6.78e+1	4.92e+2	3.42e+2	3.33e+2	4.54e+1
CoDE0	1.43e+2	8.83e+0	1.17e+2	1.77e+1	2.90e+0	1.08e+1	4.50e-5	4.30e-4	1.13e-6

SaDE exhibit the least computational costs but their reliability rates are a bit lower compared to the other algorithms.

In the problems with $D = 100$, only two algorithms (JADE and *b6e6rl*) were able to find acceptable solutions for all six problems, while the other algorithms failed at least in one problem, SaDE suffered from premature convergence in Ackley problem and Rosenbrock problem appeared too hard for two algorithms as well as Rastrigin problem. Concerning convergence speed, some algorithms outperformed JADE and *b6e6rl* in a particular problem.

Table 5 Basic characteristics of performance – average of all the problems

	$D = 30$				$D = 100$			
Alg	*ave nfe*	$rank_{nfe}$	*ave R*	$rank_R$	*ave nfe*	$rank_{nfe}$	*ave R*	$rank_R$
jDE	118082	5	99.0	4	609281	4	79.7	3
b6e6rl	74256	2	100.0	1	335411	2	99.0	1
SaDE	76377	3	92.8	6	437249	3	68.7	5
JADE	52285	1	93.8	5	203259	1	81.3	2
EPSDE	98412	4	99.8	3	733423	5	75.2	4
CoDE1	350345	7	83.3	7	1998940	7	16.7	7
CoDE0	164146	6	100.0	1	1059518	6	64.0	6

An insight to overall performance of the algorithms with respect to both the computational costs and the reliability rate is provided in Table 5, where the averaged values of the *nfe* and *R* are shown along with their ranks. Concerning the reliability rates, CoDE0 and *b6e6rl* search for an acceptable solution with $R = 100\%$ in the problems of $D = 30$ while the least average computational costs are achieved by JADE (followed by *b6e6rl* and SaDE). In the problems of $D = 100$, JADE is the best performing with respect to convergence speed followed by *b6e6rl* while they exchange their ranks when the reliability is taken into account.

In order to illustrate the differences among the algorithms in computational costs needed to reach the stopping condition and their variability, the boxplots of *nfe* are depicted in Figure 1 for the $D = 30$ and in Figure 2 for the $D = 100$, where only the runs that found an acceptable solution with respect to $f(\mathbf{x})$ (a solution with $f_{\min} < 0.0001$ in the final generation) are taken into account and CoDE1 is excluded.

The boxplots of the computational costs in Figures 1 and 2 provide us the results of successful runs of the algorithm in a given test problem. They indicate both the location and the variability. Thus we are able to see that e.g. jDE and *b6e6rl* solve Griewank problem of $D = 30$ with almost constant computational costs in all runs while the JADE and CoDE0 sometimes need *nfe* much higher than their medians of *nfe*.

The convergence speed of the algorithms can differ very significantly and the boxplots in Figures 1 and 2 show it in a very illustrative way. As expected due to the results of No Free Lunch Theorem [28], no algorithm is the fastest in all the test problems. Differences among the algorithms can be very significant like e.g. in Griewank problem of $D = 100$ or negligible like e.g. in Schwefel problem of $D = 100$, where four algorithms exhibit almost the same performance.

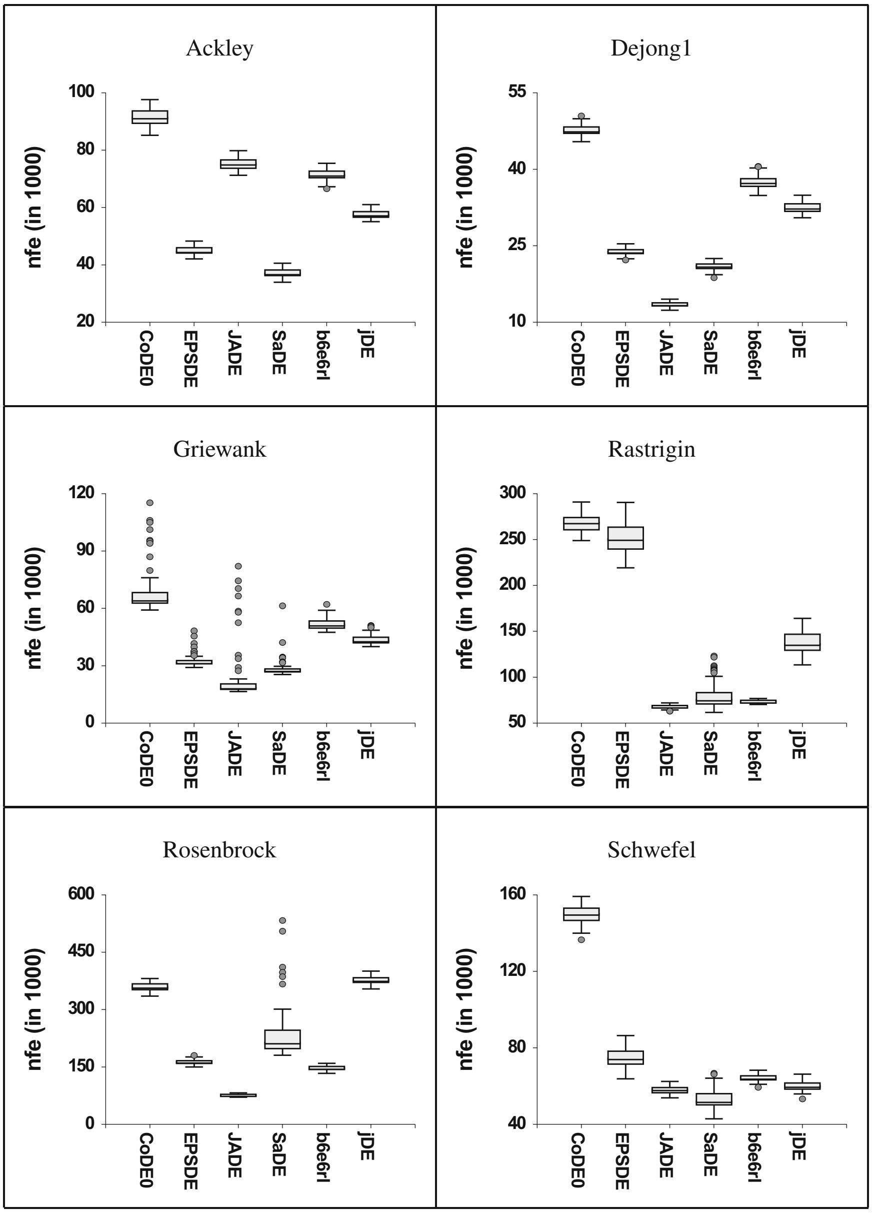

Fig. 1 Boxplots of *nfe* (only the runs that found an acceptable solution included), $D = 30$

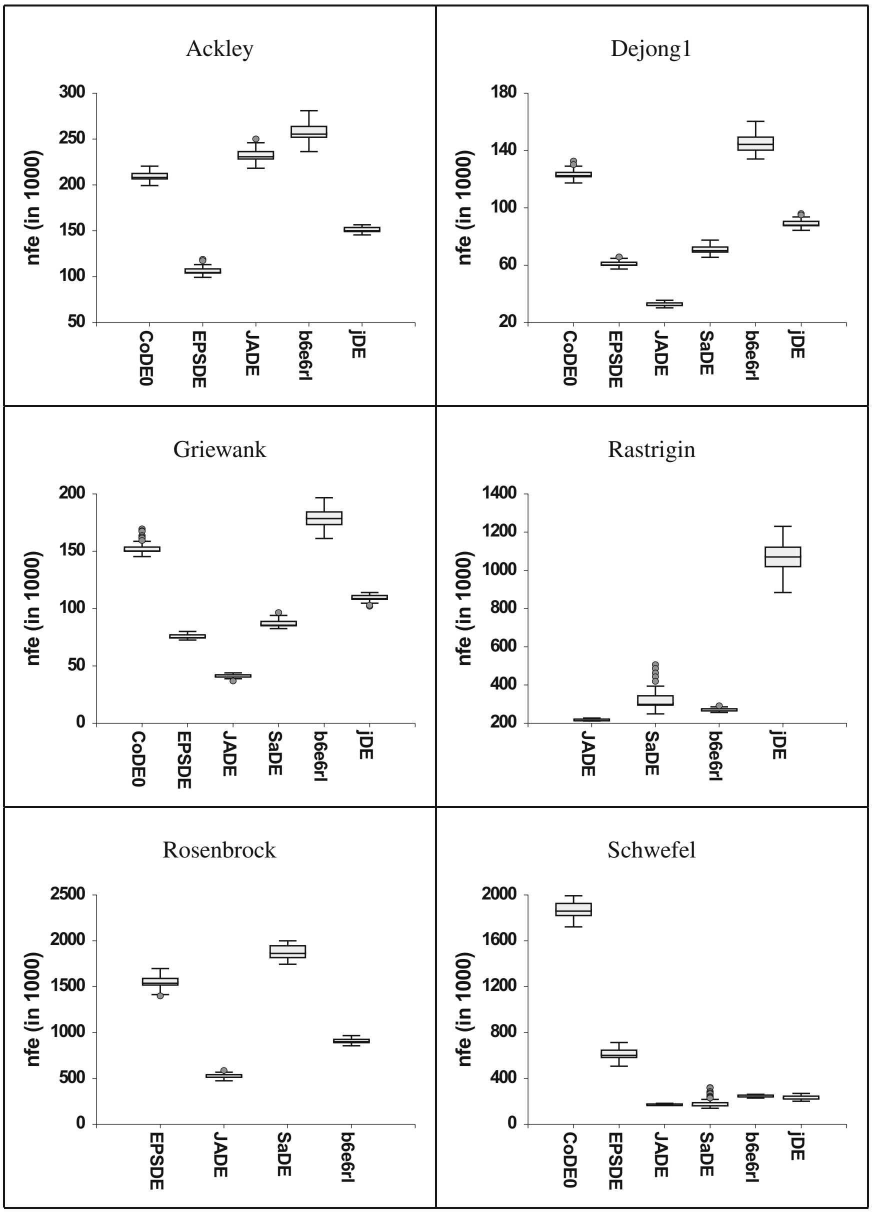

Fig. 2 Boxplots of *nfe* (only the runs that found an acceptable solution included), $D = 100$

6.2 Influence of Opposition-Based Optimization

The results of the DE variants with the application of opposition-based optimization are presented in this section. Reliability rates (R) and mean values of the *nfe* of the tested DE variants with OBO are shown for the problems with $D = 30$ and $D = 100$ in Tables 6 and 7, respectively.

The results with basic characteristics of computational costs and reliability rates in Tables 6 and 7 show similar differences in the performance of DE variants like the results of variants without OBO presented in Subsection 6.1. It can be seen at first look that the reliability rates of the variants with OBO are a bit lower, especially for the problems of $D = 100$.

Table 6 Basic characteristics of performance when OBO is applied, $D = 30$

	Ackley		Dejong1		Griewank		Rastrigin		Rosenbrock		Schwefel	
	nfe	*R*	*nfe*	*R*	*nfe*	*R*	*nfe*	*R*	*nfe*	*R*	*nfe*	*R*
jDE	56056	99	31337	100	43106	100	121412	19	600016	0	62975	78
b6e6rl	79026	100	41469	100	57994	95	120531	100	159046	100	81597	100
SaDE	37640	63	23475	100	32007	81	96847	57	255433	100	66649	95
JADE	97078	100	13978	100	21624	95	78221	100	83706	99	65660	93
EPSDE	36601	100	19411	100	29497	95	299707	99	403434	95	84043	99
CoDE1	264830	100	139135	100	207124	99	320071	99	600075	1	240045	100
CoDE0	77859	100	42079	100	60108	96	224456	92	392964	100	131050	99

Table 7 Basic characteristics of performance when OBO is applied, $D = 100$

	Ackley		Dejong1		Griewank		Rastrigin		Rosenbrock		Schwefel	
	nfe	*R*	*nfe*	*R*	*nfe*	*R*	*nfe*	*R*	*nfe*	*R*	*nfe*	*R*
jDE	151468	100	88592	100	109041	93	397891	0	2000053	0	214507	0
b6e6rl	218836	100	126343	100	158042	100	446792	25	899411	94	313928	40
SaDE	88534	1	75128	100	94835	63	281090	0	2000054	0	191269	0
JADE	303122	100	36572	100	46352	65	280138	99	622931	58	221369	66
EPSDE	77633	14	59054	100	74415	60	2000053	0	2000055	0	771732	97
CoDE1	1331498	100	755426	100	945675	100	2000118	0	2000112	0	1469938	56
CoDE0	172693	99	98936	100	126921	90	1317748	0	2000119	0	656868	34

The influence of applying the opposition-based optimization into adaptive DE variants was also compared statistically. The agreement of the computational costs measured by the number of the function evaluations was tested by the Wilcoxon two-sample (sum of ranks) test. The agreement in reliability rates was tested by the Fisher exact test for 2-by-2 contingency table. The results of two-tail tests are considered in all the comparisons. In the tables with the results of the statistical tests, the symbol "+" means better performance of DE variant with opposition-based optimization, and the symbol "−" worse performance of this DE variant. The corresponding cell of the table is left empty if no significant difference is detected at the level of $\alpha = 0.05$.

The comparison of the number of function evaluations by Wilcoxon test is presented in Table 8 for $D = 30$ and in Table 9 for $D = 100$, along with the relative changes (in %) of the number of function evaluations due to the application of OBO in the columns labeled by Δnfe. The relative change is evaluated as follows:

$$\Delta nfe = \frac{nfe_{\text{OBO}} - nfe_{\text{noOBO}}}{nfe_{\text{noOBO}}} \times 100\,, \tag{18}$$

where nfe_{OBO} and nfe_{noOBO} are the average numbers of function evaluations with and without OBO, respectively. Note that the negative value of Δnfe means better convergence (with respect to average *nfe*) of the variant with OBO and therefore, an increase of the performance.

Table 8 Relative change of *nfe* (in %) due to OBO and its significance by Wilcoxon test, $D = 30$

	Ackley		Dejong1		Griewank		Rastrigin		Rosenbrock		Schwefel	
	Δnfe	*Wilcox*	*Δnfe*	*Wilcox*	*Δnfe*	*Wilcox*	*Δnfe*	*Wilcox*	*Δnfe*	*Wilcox*	*Δnfe*	*Wilcox*
jDE	−3	+	−4	+	−1	+	−12	+	59	−	5	−
b6e6rl	11	−	11	−	12	−	64	−	8	−	27	−
SaDE	9	−	12	−	13	−	21	−	6	−	26	−
JADE	29	−	4	−	−5	−	15	−	10	−	13	−
EPSDE	−18	+	−19	+	−9	+	19	−	147	−	13	−
CoDE1	−22	+	−22	+	−25	+	−22	+	0		−20	+
CoDE0	−15	+	−12	+	−12	+	−16	+	9	−	−13	+

Table 9 Relative change of *nfe* (in %) due to OBO and its significance by Wilcoxon test, $D = 100$

	Ackley		Dejong1		Griewank		Rastrigin		Rosenbrock		Schwefel	
	Δnfe	*Wilcox*	*Δnfe*	*Wilcox*	*Δnfe*	*Wilcox*	*Δnfe*	*Wilcox*	*Δnfe*	*Wilcox*	*Δnfe*	*Wilcox*
jDE	0.1		−0.7	+	−0.6		−63	+	0		−8	+
b6e6rl	−15	+	−13	+	−12	+	65	−	−1	+	27	−
SaDE	14	−	6	−	9	−	−13	+	6	−	6	−
JADE	30	−	11	−	12	−	28	−	20	−	27	−
EPSDE	−24	+	−3	+	−2	+	0		29	−	26	−
CoDE1	−33	+	−62	+	−53	+	0		0		−27	+
CoDE0	−17	+	−20	+	−17	+	−34	+	0		−65	+

The influence of OBO on the computational costs was detected frequently but it appears both in the positive and the negative way. For the problems of $D = 30$, the application of OBO leads to significant decrease of *nfe* in 17 out of the 42 cases and to significant increase of *nfe* in 24 cases. Significant increase of *nfe* was detected in 15 cases, significant decrease in 20 cases for the problems of $D = 100$. Moreover, some differences in *nfe* were found statistically significant even in the cases where the relative change Δnfe is negligible from practical point of view, i.e. $|\Delta nfe| \le 1\%$.

The comparison of the reliability rates by Fisher test is shown in Table 10 for $D = 30$ and in Table 11 for $D = 100$, where the differences in the values of R achieved due to the OBO are also presented in the columns labeled ΔR. The results in Tables 10 and 11 indicate that the influence of OBO on the reliability rate is not very strong. No significant change in the reliability was observed in 36 cases out of 42 for the problems of $D = 30$ and in 28 cases out of 42 for the problems of $D = 100$. Positive influence was found only in 4 cases (3 out of them in the worst performing CoDE1, see Table 11) and negative one in 16 cases out of the total 84 cases.

Table 10 Differences of R due to OBO and significance of changes by Fisher test, $D = 30$

	Ackley		Dejong1		Griewank		Rastrigin		Rosenbrock		Schwefel	
	ΔR	*Fisher*	ΔR	*Fisher*	ΔR	*Fisher*	ΔR	*Fisher*	ΔR	*Fisher*	ΔR	*Fisher*
jDE	−1		0		0		−79	−	−97	−	−21	−
b6e6rl	0		0		−5		0		0		0	
SaDE	−12		0		−6		−43	−	5		−5	
JADE	0		0		2		0		6		16	+
EPSDE	0		0		−5		−1		−5		0	
CoDE1	0		0		−1		−1		1		0	
CoDE0	0		0		−4		−8	−	0		−1	

Table 11 Differences of R due to OBO and significance of changes by Fisher test, $D = 100$

	Ackley		Dejong1		Griewank		Rastrigin		Rosenbrock		Schwefel	
	ΔR	*Fisher*	ΔR	*Fisher*	ΔR	*Fisher*	ΔR	*Fisher*	ΔR	*Fisher*	ΔR	*Fisher*
jDE	0		0		−6		−92	−	0		−87	−
b6e6rl	0		0		1		−75	−	−3		−58	−
SaDE	1		0		−7		−91	−	−51	−	−100	−
JADE	0		0		0		−1		−11		12	
EPSDE	−75	−	0		−22	−	0		−82	−	1	
CoDE1	100	+	0		100	+	0		0		56	+
CoDE0	2		0		2		0		0		−65	−

We can conclude that the application of OBO to adaptive versions of differential evolution does not bring such benefit as it was reported in the case of standard DE variants [17]. The usage of OBO in adaptive DE variants increases the reliability rate rarely. Convergence rate is increased by OBO sometimes, on other hand the OBO causes deterioration of the convergence in a part of the test problems.

6.3 Other Characteristics of Search Process

The rate of the successful trial-vector generation (i.e. the proportion of the trial vectors satisfying the condition $f(\mathbf{y}) \leq f(\mathbf{x}_i)$) is a useful characteristic of the search process. It is apparent that a high value of the rate leads to fast convergence but with the risk of premature convergence without finding the region of the global minimum due to the loss of the explorative ability. Vice versa, very low rate of the generating of successful trial vectors causes slow convergence or stagnation of the search process. The average rate values of the successful trial-vector generation expressed in percent of *nfe* are presented in Tables 12 and 13 for the problems of $D = 30$ and $D = 100$, respectively. In order not to overload the reader by other tables, we present only the results of the algorithms without OBO in this section.

Table 12 Average rate values of the generating of successful trial vectors expressed in percent of *nfe*, $D = 30$

	Ackley	Dejong1	Griewank	Rastrigin	Rosenbrock	Schwefel
jDE	25	26	25	9	13	19
b6e6rl	28	30	29	22	27	25
SaDE	32	35	34	16	35	23
JADE	23	51	47	22	45	27
EPSDE	25	28	27	4	24	13
CoDE1	6	7	6	4	3	5
CoDE0	16	18	16	4	17	8

Table 13 Average rate values of the generating of successful trial vectors expressed in percent of *nfe*, $D = 100$

	Ackley	Dejong1	Griewank	Rastrigin	Rosenbrock	Schwefel
jDE	19	19	19	3	15	10
b6e6rl	26	26	26	20	27	23
SaDE	33	36	36	35	37	21
JADE	18	58	58	19	52	23
EPSDE	25	26	25	0	24	4
CoDE1	2	2	2	0	0	1
CoDE0	16	17	17	0	17	2

The rate of the successful trial vectors generations is highly negatively correlated with *nfe*, the value of Pearson correlation coefficient is -0.65 for the problems of $D = 30$ and -0.59 for the problems of $D = 100$, the both values of the correlation coefficient are significantly different from zero, $p < 5 \times 10^{-7}$ and the null hypotheses on the zero correlation are rejected at any reasonable level of significance. Hence to follow up the current values of this rate could be helpful for the decision on the efficiency of an algorithm in a certain problem. The low values of the average rate of four algorithms in Rastrigin problem of $D = 30$ explain their high computational costs needed for reaching the stopping condition.

Another characteristic of the search process is the proportion of *nfe* needed to find an acceptable solution of the problem by an algorithm. The solution found in a run is considered acceptable in these tests if the minimum function value in the final generation does not differ from the known correct solution of the test problem by more than 1×10^{-4}. In Tables 14 and 15, the average percentage of *nfe* required to find an acceptable solution is presented. If no acceptable solution of the problem is found by an algorithm, the corresponding cell is empty.

Table 14 Average percentage of *nfe* required to find an acceptable solution, $D = 30$

	Ackley	Dejong1	Griewank	Rastrigin	Rosenbrock	Schwefel
jDE	74	71	79	93	91	84
b6e6rl	77	72	80	86	92	84
SaDE	76	75	80	92	95	88
JADE	85	73	75	89	94	87
EPSDE	76	72	80	97	93	91
CoDE1	76	71	81	87		83
CoDE0	77	73	81	95	94	91

Table 15 Average percentage of *nfe* required to find an acceptable solution, $D = 100$

	Ackley	Dejong1	Griewank	Rastrigin	Rosenbrock	Schwefel
jDE	78	77	81	98		91
b6e6rl	80	77	82	88	96	87
SaDE	69	82	86	96	99	92
JADE	95	81	85	96	98	96
EPSDE	82	80	84		99	98
CoDE1		78				
CoDE0	82	80	84			99

The proportions of *nfe* needed to find an acceptable solution vary both with respect to the algorithms and the functions. The values of the proportion are lower for easy problems like Ackley and Dejong1, where all the algorithms are able to find an acceptable solution in about $3/4$ of *nfe*. For hard Rosenbrock problem, they need more than 90 % but once an acceptable solution is found, the whole population is concentrated to the global minimum very quickly.

7 Source Codes of DE Algorithms

Source code of the algorithms tested in this study is available online[1]. The adaptive DE variants were implemented by the authors of this chapter according to their descriptions in the journal papers cited in Section 5. All the algorithms are implemented in Matlab except JADE which is written in C. The versions of the algorithms used in test problems are provided. Hence the codes have some parts to follow up some characteristics of the search process used in comparison of the algorithms in benchmark problems but these are not necessary in real-world applications. Nevertheless, these parts of codes and the corresponding input/output parameters do not prevent from the use of the implemented algorithms to other problems. Examples of calling the functions (modules) implementing the adaptive DE variants are also provided for each algorithm, including the settings the auxiliary input parameters controlling the adaptation to their recommended values used in these experiments.

8 Recommendations for Applications

The adaptive DE variants described above are easy to apply in the solution of the real-world optimization problem. The users need to implement their objective function, define the stopping condition (usually to set up the values of ε_f and *maxevals* in (17) is sufficient), and set up the size of the population. In the case of adaptive DE variants, the population size can be smaller than the values recommended for non-adaptive DE strategies. In this study, the population size was set to $NP = 60$ for all the test problems both of $D = 30$ and $D = 100$, which appears reasonable.

When solving a real-world optimization problem without a priori knowledge about the property of the objective function, the recommendations bellow should be followed.

1. Form the stopping condition (17) by setting its input parameters. The value of ε_f depends on the problem to be solved, usually $\varepsilon_f \leq 1 \times 10^{-6}$ is reasonable. If it is known a priori that $f(\mathbf{x}) > 0, \forall \mathbf{x} \in \Omega$, the stopping condition can be formed in the term of the relative difference:
$$\frac{f_{\max} - f_{\min}}{f_{\min}} < \varepsilon_f \, .$$
The value of *maxevals* in the range of $[1 \times 10^4, \; 1 \times 10^5]$ seems to be sufficient for adaptive DE variants.
2. Set up the size of population. A proper value of *NP* depends on the dimension of the problem. When adaptive DE variants are used, a good choice is $NP \geq 20$ for the problems of small dimension and $50 \leq NP \leq 3 \times D$ for the high-dimensional problems.
3. Do not rely on the result obtained by one algorithm! No heuristic search can guarantee that a good approximation of the global minimum is found in the

[1] `http://www1.osu.cz/~tvrdik/down/global_optimization.html`

finite number of search steps. Hence the application of several adaptive variants is recommendable when solving a problem without any a priori knowledge. If the results found by the used variants do not differ significantly, they can be considered as a good approximation of the global minimum. Otherwise, a variant giving the best results should be applied in several repetitions. The solution found can be considered acceptable if the results obtained from the repetitions agree. If the agreement of the repeated results is not achieved, the whole procedure should be restarted with a new setting of its input parameters, e.g. the increase of *NP* twice might be helpful in such a case and the procedure continues by step 3.

The recommendations formed above can be implemented into a computer program and applied automatically if a reliable solution of the problem is needed. The fast convergence of adaptive DE variants enables the application of several algorithms in the solution of a problem with a smaller effort and lower computational costs than the trial-and-error control parameter tuning usually needed in the application of a non-adaptive DE variant.

9 Conclusion

Seven state-of-the-art adaptive variants of differential evolution were tested in benchmark problems of two levels of dimension. Moreover, the versions of all the algorithms with the implementation of opposition-based optimization were also tested in the same benchmark problems and the performance of the corresponding algorithms was compared statistically.

The adaptive DE variants except CoDE1 searched for the minimum with high reliability in the problems of $D = 30$. Their minimum reliability rate is 75 % if CoDE1 failing in Rosenbrock problem is omitted. In the problems of $D = 100$, the values of the reliability rate are more dispersed. Each DE variant except JADE and *b6e6rl* failed completely in the search of an acceptable solution at least in one test problem. The most reliable DE variant is *b6e6rl* with the reliability rate of 99 %. It might be caused by the employing of the exponential crossover in a part of competing strategies as it is considered to be beneficial for the reliability of the algorithm in the solution of non-separable problems. Note that among the tested DE variants, the exponential crossover is used only in *b6e6rl*.

Considering the efficiency of the algorithms, an algorithm outperforming others in a certain problem can be found but the rank of the algorithm is problem-dependent. This observation was expected because it is in accordance with the results of No Free Lunch theorem [28]. Moreover, higher convergence rate of the algorithm is often paid by lower reliability of the search, if the algorithm is too greedy for a given problem. According to average computational costs, JADE was the most efficient algorithm in the problems of dimension either $D = 30$ or $D = 100$.

The implementation of the opposition-based optimization into adaptive DE variants does not bring the effect that could be expected based on the results achieved in non-adaptive DE [17]. Implementation of OBO increases the reliability very rarely and its influence on the efficiency is questionable. The opposition-based optimization is sometimes profitable but sometimes very disadvantageous and the count of drawbacks is comparable or even higher than the count of benefits. Possible explanation of the results may be in the balanced exploitation and exploration of the search in adaptive DE variants.

Based on the results of the experiments and conclusions in literature, the recommendations for real-world applications of adaptive DE are formed and the source codes of adaptive DE algorithms are also made available to potential users. The adaptive DE variants tested in this study do not require to set up the F and CR parameters that are crucial for the performance of standard DE in a given optimization problem. Thus, using adaptive DE variants provide the solution less time-consuming and more comfortable.

Acknowledgements. This chapter has been elaborated in the framework of the IT4Innovations Centre of Excellence project, reg. no. CZ.1.05/1.1.00/02.0070 supported by Operational Programme 'Research and Development for Innovations' funded by Structural Funds of the European Union and state budget of the Czech Republic. The work was also supported by University of Ostrava from the SGS 22/2011-2 project.

References

1. Ali, M.M., Törn, A.: Population set based global optimization algorithms: Some modifications and numerical studies. Computers and Operations Research 31, 1703–1725 (2004)
2. Bäck, T.: Evolutionary Algorithms in Theory and Practice. Oxford University Press, New York (1996)
3. 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)
4. Das, S., Abraham, A., Chakraborty, U.K., Konar, A.: Differential evolution using a neighborhood-based mutation operator. IEEE Transactions on Evolutionary Computation 13(3), 526–553 (2009)
5. Das, S., Suganthan, P.N.: Differential evolution: A survey of the state-of-the-art. IEEE Transactions on Evolutionary Computation 15, 27–54 (2011)
6. Feoktistov, V.: Differential Evolution in Search of Solutions. Springer (2006)
7. Gämperle, R., Müller, S.D., Koumoutsakos, P.: A parameter study for differential evolution. In: Grmela, A., Mastorakis, N.E. (eds.) Advances in Intelligent Systems Fuzzy Systems, Evolutionary Computing, pp. 293–298. WSEAS Press, Athens (2002)
8. Kaelo, P., Ali, M.M.: A numerical study of some modified differential evolution algorithms. European J. Operational Research 169, 1176–1184 (2006)
9. Liu, J., Lampinen, J.: A fuzzy adaptive differential evolution algorithm. Soft Computing 9, 448–462 (2005)
10. 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)
11. Neri, F., Tirronen, V.: Recent advances in differential evolution: a survey and experimental analysis. Artificial Intelligence Review 33, 61–106 (2010)
12. Omran, M.G.H., Salman, A., Engelbrecht, A.P.: Self-adaptive Differential Evolution. In: Hao, Y., Liu, J., Wang, Y.-P., Cheung, Y.-m., Yin, H., Jiao, L., Ma, J., Jiao, Y.-C. (eds.) CIS 2005. LNCS (LNAI), vol. 3801, pp. 192–199. Springer, Heidelberg (2005)

13. Price, K.V.: An introduction to differential evolution. In: New Ideas in Optimization, pp. 293–298. McGraw-Hill, London (1999)
14. Price, K.V., Storn, R., Lampinen, J.: Differential Evolution: A Practical Approach to Global Optimization. Springer (2005)
15. Qin, A., Huang, V., Suganthan, P.: Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Transactions on Evolutionary Computation 13, 398–417 (2009)
16. Qin, A.K., Suganthan, P.N.: Self-adaptive differential evolution for numerical optimization. In: IEEE Congress on Evolutionary Computation, pp. 1785–1791 (2005)
17. Rahnamayan, S., Tizhoosh, H.R., Salama, M.M.A.: Opposition-based differential evolution. IEEE Transactions on Evolutionary Computation 12, 64–79 (2008)
18. Rahnamayan, S., Tizhoosh, H.R., Salama, M.M.A.: Opposition-based differential evolution. In: Chakraborty, U.K. (ed.) Advances in Differential Evolution, pp. 155–171. Springer (2008)
19. Ronkkonen, J., Kukkonen, S., Price, K.V.: Real-parameter optimization with differential evolution. In: IEEE Congress on Evolutionary Computation, pp. 506–513 (2005)
20. Shang, Y.W., Qiu, Y.H.: A note on the extended Rosenbrock function. Evolutionary Computation **14**(1), 119–126 (2006)
21. Storn, R., Price, K.V.: Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. Int. Comp. Sci., Inst., Berkeley, TR-95-012 (1995), `http://www.icsi.berkeley.edu/~storn/litera.html`
22. 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)
23. 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/`
24. Tvrdík, J.: Competitive differential evolution. In: Matoušek, R., Ošmera, P. (eds.) 12th International Conference on Soft Computing, MENDEL 2006, pp. 7–12. University of Technology, Brno (2006)
25. Tvrdík, J.: Adaptation in differential evolution: A numerical comparison. Applied Soft Computing 9(3), 1149–1155 (2009)
26. 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`
27. 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)
28. Wolpert, D.H., Macready, W.: No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation 1, 67–82 (1997)
29. 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)
30. Zaharie, D.: Influence of crossover on the behavior of differential evolution algorithms. Applied Soft Computing 9, 1126–1138 (2009)
31. Zhang, J., Sanderson, A.C.: Adaptive Differential Evolution: A Robust Approach to Multimodal Problem Optimization. Springer (2009)
32. Zhang, J., Sanderson, A.C.: JADE: Adaptive differential evolution with optional external archive. IEEE Transactions on Evolutionary Computation 13, 945–958 (2009)