On Modification of Population-Based Approach Used in Adaptive Differential Evolution Algorithm

Petr Bujok

Department of Computer Science, University of Ostrava
30. dubna 22, 70103 Ostrava, Czech Republic
petr.bujok@osu.cz

A new approach for the mutation operation in the differential evolution (DE) algorithm is introduced. The aim of this technique is to enhance the mutation strategy to avoid the local minimum area. The proposed method is implemented to five state-of-the-art DE variants and the standard DE variant DE/rand/1/bin. Twelve DE variants are compared on CEC 2015 problems at four dimension levels. The results show that the proposed method is able to increase the performance of the original DE variants in the significant part of the test problems.

Keywords: Global optimization problem; differential evolution; auxiliary population; experimental comparison; CEC 2015 test suite

1 Introduction

A single-objective global optimization problem with bound constraints is defined as follows. The cost function to be minimized is $f(\vec{x})$, $\vec{x} = (x_1, x_2, \ldots, x_D) \in \mathbb{R}^D$. The domain of feasible solutions Ω is constrained by bounds, a lower limit (a_j) and an upper limit (b_j), $\Omega = \prod_{j=1}^{D}[a_j, b_j]$, $a_j < b_j$, $j = 1, 2, \ldots, D$. The global minimum point $\vec{x}^*$ satisfying condition $f(\vec{x}^*) \leq f(\vec{x})$, $\forall \vec{x} \in \Omega$ is the solution of the problem. The fitness is inversely proportional to the cost function, in the case of the minimization problem.

There is no deterministic algorithm which solves the global optimization problem in a polynomial time in general. Evolutionary algorithms are mostly able to provide an acceptable solution in a reasonable computational time. However, even very sophisticated adaptive evolutionary algorithms cannot guarantee the finding of an acceptable solution in the finite computational time. There are optimization problems, where the state-of-the-art evolutionary algorithms fail. That is why new concepts applied to evolutionary algorithms are intensively studied. Differential evolution (DE) is one of the leading paradigms among the evolutionary algorithms.

In this paper, a new approach for the mutation operation in the DE algorithm is introduced. This approach is focused on the problem when the optimization algorithms

get stuck in the local minimum. The newly proposed approach in the operation of mutation is applied in several well known DE or adaptive DE variants. Selected DE variants and the corresponding counterparts are used to solve the problems of CEC 2015 test suite.

The rest of the paper is organized in the following manner. The basic scheme of the DE algorithm, a brief description of the selected DE variants and some related works are shown in Section 2, 2.1 and 2.2. The new approach for DE mutation is proposed in Section 3. Settings of experiments and their results are given in Section 4 and 5. Finally, conclusions are made in the last Section.

2 Differential Evolution Algorithm

Differential evolution introduced by Storn and Price in [11] is a population-based evolutionary algorithm for problems with a real-valued cost function. The population P of the size N is developed step-by-step from generation to generation.

DE uses evolutionary operators, i.e. mutation, crossover, and selection that are applied in the development of a new generation of P. The DE algorithm is shown in a pseudo-code in Algorithm 1.

Algorithm 1 Differential evolution algorithm

```
initialize population P = {x_1, x_2, ..., x_N}
evaluate f(x_i),    i = 1, 2, ..., N
while stopping condition not reached do
  for i = 1, 2, ..., N do
    create a new trial vector y (mutation and crossover)
    evaluate f(y)
    if f(y) ≤ f(x_i) then
      insert y into Q
    else
      insert x_i into Q
    end if
  end for
  P ← Q
end while
```

The new trial point $\vec{y}$ is created from a mutant point $\vec{u}$ generated by using a kind of the mutation and from the current point of the population $\vec{x}_i$ by the application of the crossover. A combination of the mutation and the crossover variant is usually called a DE strategy. The mutation can cause that a mutant point $\vec{u}$ moves out of the domain Ω. In such a case, the values of $u_j \notin [a_j, b_j]$ are turned over into Ω by using transformation $u_j \leftarrow 2 \times a_j - u_j$ or $u_j \leftarrow 2 \times b_j - u_j$ for the violated component. A better point from the pair of $\vec{x}_i$, $\vec{y}$, based on the value of the cost function, is selected for the new generation (Q).

The most frequently used mutation strategy in DE is rand/1 and it is generated as follows:

$$\vec{u} = \vec{x}_{r1} + F(\vec{x}_{r2} - \vec{x}_{r2}) \tag{1}$$

where $\vec{x}_{r1}, \vec{x}_{r2}, \vec{x}_{r3}$ are three mutually distinct points taken randomly from population P, not coinciding with the current $\vec{x}_i$, and $F > 0$ is a control parameter. The crossover operator constructs the trial vector $\vec{y}$ from current individual $\vec{x}_i$ and the mutant vector $\vec{u}$. There are two types of a crossover and a binomial crossover replaces the elements of vector $\vec{x}_i$ using the following rule:

$$\vec{y}_{i,j} = \begin{cases} \vec{u}_{i,j} & \textit{if } rand_j(0,1) \leq CR \textit{ or } j = rand(1,D) \\ \vec{x}_{i,j} & \textit{otherwise}. \end{cases} \tag{2}$$

where $rand(1,D)$ is random vector uniformly distributed in $[0,1)$ and $CR \in [0,1]$ is a control parameter influencing the number of elements to be exchanged by the crossover. Eq. (2) ensures that at least one element of $\vec{x}_i$ 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.

The DE algorithm has several control parameters whose settings significantly influence the ability to solve different optimization problems. These control parameters and their settings have been intensively studied in recent years. A comprehensive summary of an advanced results in DE research is available in [2, 3, 7], where several kinds of the mutation and the crossover were listed and some adaptive or self-adaptive DE variants were described. Swagantan et al.proposed a wide survey of state of the art of differential evolution algorithm. [4]. No strategy, i.e. a combination of a mutation and crossover variant, is able to outperform all remaining strategies in the case of all optimization problems. This fact corresponds with the so-called No-free-lunch theorem [15]. On the other hand, adaptive variants of DE enable to change DE control parameters during the run of the algorithm to the current problem without trial-and-error tuning of the control parameters [9, 16].

2.1 Adaptive DE Variants

The self-adaptive DE variants (jDE [1], JADE [18], SaDE [10], and EPSDE [6]) are currently considered 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 variants are also included in this experiment along with a variant of composite trial vector generation strategies and the control parameters that have been recently published (CoDE) [12].

A simple and efficient adaptive DE variant (mostly called jDE in literature) was proposed by Brest et al. [1]. It uses the DE/rand/1/bin with an evolutionary self-adaptation of F and CR. The pair 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 the next generation. The values of F and CR are initialized randomly for each point $\vec{x}_i$ in population P 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.

The differential evolution algorithm with strategy adaptation (SaDE) was introduced by Qin and Suganthan. A more sophisticated and a more efficient variant was proposed later in [10] and it is used in our experimental comparison. Four mutation strategies (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. Each strategy has a probability to be selected and applied and these probabilities are updated after each LP generations.

JADE is an algorithm of the adaptive differential evolution, introduced by Zhang and Sanderson in [18]. The original DE concept is extended with three different improvements - current-to-pbest mutation strategy, adaptive control of parameters F and CR, and archive A. The archive A is initialized as an empty set. In every generation, parent individuals are replaced by a better offspring, $(f(\vec{y}) \leq f(\vec{x}_i))$, individuals are put into the archive. After every generation the archive size is reduced to N individuals by randomly dropping surplus individuals.

In the EPSDE adaptive variant [6], 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 of the population. If the parent vector produces a successful offspring vector, this triplet survives with the trial vector for the next generation and it is also stored. Otherwise, the triplet is randomly reinitialized.

The last adaptive DE variant used in the experiments is DE with composite trial vector generation strategies and control parameters, CoDE, presented by Wang et al. [12]. 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 (select the offspring vector with the least function from the triplet).

2.2 Related Works

Zamuda and Brest [17] introduced a detail analysis of controlling of the F, CR parameters in a new adaptive DE variant (SPSRDEMMS) derived from jDE [1]. Multi-mutation strategy mechanism is applied to the population size reduction. Applied population-size reduction mechanism decreases a population size to half of a certain size in three defined stages. Moreover, the population of individuals is divided into two sub-populations, superior and inferior ones, with respect to the

function values. Further, a migration of the best individual from the better part to the worse part is occasionally performed. The analysis shows the influence of very small changes of τ_1 and τ_2 on the efficiency of SPSRDEMMS.

Wang et al. proposed in 2016 [13] a new DE variant with enhanced covariance matrix crossover called CPI-DE. In this algorithm, a covariance matrix is computed using the mean vector of the search distribution and it is further updated after each generation. This covariance matrix enables to generate new individuals in the Eigen coordinate system (principal components). Two new individuals are generated for each parent-vector, one in the standard coordinate system and the second one in Eigen coordinate system. The best one from triplet of parent, children1 and children2 is used in next generation. CPI-DE is able to increase performance in two classic DE variant and three state-of-the-art algorithms as it shown on CEC 2013 and CEC2014 test suites at dimension levels $D = 30$ and $D = 50$. CPI-JADE variant is further compared with three various JADE versions applying another covariance-matrix approaches.

Wang et al. study a restrained condition of selecting individuals for mutation in DE [14]. This condition ensures that a randomly selected individuals for mutation are mutually different and that there is no degenerated (zero-valued) differential vector. Authors performed experimental comparison of this restrained condition in six classic DE variants with various mutation and seven state-of-the-art DE algorithm on two benchmark sets (CEC 2005 and CEC 2013). Results show that three out of six classic DE variants perform significantly better when restrained condition is violated. In the case of adaptive DE, only CoDE variant without this restrained condition perform significantly better than the original CoDE algorithm.

Piotrowski in [8] summarized a population size control. Author widely compared several fixed and flexible population size setting in ten various DE algorithms on two various test benchmark sets. Author recommended fixed population size $N = 100$ for artificial problems with smaller dimensions, for middle dimension (approximately $D = 30$) a fixed settings $N = 3D$ or $N = 5D$ are the best choice. For dimension $D = 50$, the best performing population size was found $N = 5D$ or $N = 100$. For the real-world problems with various dimensionality ($1 \leq D \leq 240$) the population size should be set $N = 100$ or $N = 50$. In the case of flexible population size, JADE with reduction of N based on (un-)successes in previous generations was the best performing algorithm. If there is no improvement in 4 generations, 1 % of poorer individuals are removed, and vice versa, if the population is improved, 1 % of newly generated individuals are added.

3 Enhanced Approach for DE Mutation Strategy

A new approach to increase the efficiency of DE algorithm is proposed. The main idea is to enhance the mutation strategy to avoid the local minimum area. A significant problem is that when the population contains some potentially good solutions the population is slowly moved towards these good individuals. Sometimes this fact causes that the solution is detected quickly, but often only a poor local solution is

found. The solution to avoid the population from getting stuck in the local solution area could be in our new DE mutation strategy approach.

Algorithm 2 Differential evolution with enhanced mutation strategy

```
initialize population P = {x_1, x_2, ..., x_N}
evaluate f(x_i),   i = 1, 2, ..., N
initialize auxiliary population R = {z_1, z_2, ..., z_Nr}, Nr = N * pr, pr ∈ [0, 1]
while stopping condition not reached do
  for i = 1, 2, ..., N do
    create a mutation vector u from P and R (using 3 rules)
    produce a new trial point y
    evaluate f(y)
    if f(y) ≤ f(x_i) then
      insert y into Q
    else
      insert x_i into Q
      if point from R was used then
        reinitialize currently used point from R
      end if
    end if
  end for
  P ← Q
end while
```

Besides the population P of the N potential solutions another population R is initialized and kept in the initialized form during the search process. Sometimes, one of the vectors in the mutation strategy could be selected from the auxiliary population R, to leave the local solution area.

The situation in Figure 1 illustrates when the points of the population P (circles filled white) get stuck in the local solution area. In spite of the stuck P, the points of the auxiliary population R (circles filled black) are located out from the local solution area and they could move the points of P.

This mutation approach has several parameters whose settings significantly influence the efficiency of this method. The first parameter is the size of the auxiliary population R, denoted Nr. The value of this parameter should be taken from the interval $[1,N]$. A bigger size of R means more places to move in the area Ω. We suppose to compute the value of Nr as a proportion of the population P, $Nr = N \cdot pr$, where pr is a real number from $[0,1]$.

The next parameter specifies which point(s) of the actually used mutation strategy will be selected from Nr. We suppose a very simple idea based on three rules:

1. only one point in the mutation strategy could be selected from R,
2. if the best point is used in the mutation strategy, it is never taken from R,
3. the first point in the equation of the mutation strategy is never taken from R.

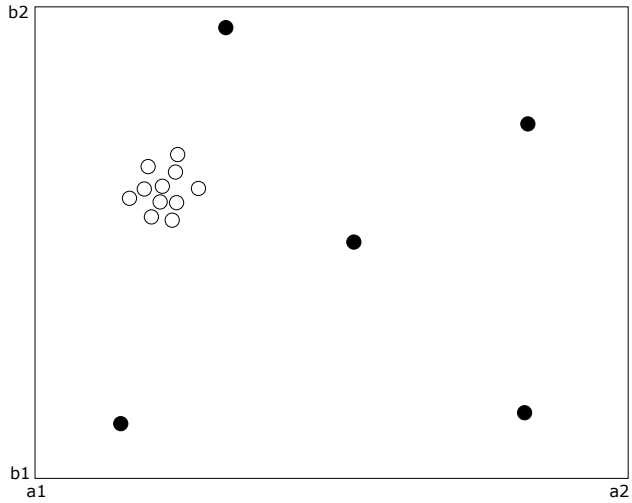

Figure 1
Population located in the local solution area and point of the auxiliary population R

For example, in rand/1 mutation strategy (1) only one of the points in bracket ($\vec{x}_{r2}$ or $\vec{x}_{r3}$) could be taken from R.

The last setting of the proposed method specifies how often the auxiliary population is used. There are many possible ways how to set this parameter, we suppose the simplest one. The point in the mutation strategy which is given to be from R is selected from the union of both populations $P \cup R$ in each step of each generation. It means that only the size of the auxiliary population, pr, could control the frequency of using points from R. The bigger the pr value is, the more frequently points from R are used and vice versa.

When the point $\vec{z}_j$ from the auxiliary population R was used and generate a successful trial individual $(f(\vec{y}) \leq f(\vec{x}_i))$, then $\vec{z}_j$ survives in R. On the other hand, when a new trial individual is generated using some point of R and it is unsuccessful $(f(\vec{y}) > f(\vec{x}_i))$, the used point from R is randomly reinitialized in Ω to move to a better place.

4 Experiments and Settings

The aim of the experiments in this paper is to verify the performance of the proposed enhanced approach in mutation strategy. We apply the proposed method in six DE variants, one standard DE with fixed settings and five adaptive DE variants mentioned in Section 2.1. All algorithms in the experiment are implemented in the Matlab environment.

The test suite of 15 problems was proposed for a special session on Real-Parameter Numerical Optimization, a part of the Congress on Evolutionary Computation (CEC)

2015. This session was intended as a competition of optimization algorithms where new variants of algorithms are introduced. The functions are described in [3] including the experimental settings required for the competition. The source code of the functions is also available on the web site given in this report.

It is expected that CEC test suite will currently become one of the most relevant benchmarks required for publishing new single-objective optimization algorithms. The test functions CEC 2015 are divided into four categories, based on its difficulty (from easy to hard): unimodal functions (F1, F2), multimodal functions (F3 - F5), hybrid functions (F6 - F8) and composition functions (F9 - F15).

Our tests were carried out at four levels of dimension, $D = 10, 30, 50, 100$, with 51 independent runs per each test function. Each point in the population is evaluated by the cost function. The function-error value is computed as the difference between the function value of the current point and the known function value in the global minimum point. The run of the algorithm stops if the prescribed amount of function evaluation $MaxFES = D \times 10^4$ is reached or if the minimum function error in the population is less than 1×10^{-8}. Such an error value is considered sufficient for an acceptable approximation of the correct solution. The search range (domain) for all the test functions is $[-100, 100]^D$.

The parameters of the state-of-the-art DE variants are set to the recommended values. The values of the jDE variant parameters, $\tau_1 = 0.1$, $\tau_2 = 0.1$, the learning period of the SaDE variant, $LP = 50$, the learning period of the EPSDE variant, $LP = N = 100$. The population size of P is set to $N = 100$ for state-of-the-art algorithms and $N = 30$ for standard DE/rand/1/bin variant. The control parameters of F and CR are in standard DE set $F = 0.8$, $CR = 0.8$. The size of the auxiliary population R is set $Nr = N \times pr$, where $pr = 0.05$. Then the value is $Nr = 5$ for the state-of-the-art variants and $Nr = 2$ for the standard DE/rand/1/bin variant.

5 Results

The median values of twelve compared algorithms (six original DE variants and six corresponding counterparts based on the proposed approach) are presented in Tables 3-14. The original variants are denoted based on the abbreviation mentioned above and the names of the new DE variants are enhanced by the text -mut. The median values were computed from 51 independent runs for each algorithm, function and dimension level. The median value for a better variant from the pair original-new is printed in bold.

The performance of the proposed method is compared by Wilcoxon two-sample test. The results of these tests are shown in the column Sign.. The symbol + denotes that new approach outperforms the original DE variant significantly, the symbol - marks better performance of the original DE variant and the symbol $\approx$ is used when there is no significant difference between the DE variants. When the DE variant of the pair is significantly better, the median value is printed in bold and underlined.

For a better comparison of the performance of the proposed approach, the number

of the wins and losses are counted in Table 1. These numbers are counted independently for each pair of algorithm and each dimension level. Based on the percent values of wins we can observe that the enhanced mutation method has the least efficiency for the JADE (30 %) and EPSDE (33 %) variants. Contrary, the most efficiency of the new approach is for the standard DE (62 %), for jDE (48 %) and CoDE (47 %) variants. The overall performance of all 12 algorithms was compared

Table 1
The number of wins and losses of adaptive DE variants and corresponding counterparts

Alg.	$D = 10$	$D = 30$	$D = 50$	$D = 100$	$\sum$	%
CoDE	5	5	7	6	23	38.3
CoDE-mut	6	8	6	8	28	**46.7**
DE	5	6	3	4	18	30
DE-mut	7	8	11	11	37	**61.7**
EPSDE	7	7	7	8	29	**48.3**
EPSDE-mut	4	5	6	5	20	33.3
JADE	4	8	7	12	31	**51.7**
JADE-mut	5	5	6	2	18	30
jDE	4	5	5	5	19	31.7
jDE-mut	7	8	7	7	29	**48.3**
SaDE	4	8	5	7	24	40
SaDE-mut	6	5	8	8	27	**45**

Table 2
Mean ranks from Friedman-rank test results for all the algorithms in comparison

Alg.	$D = 10$	$D = 30$	$D = 50$	$D = 100$	Mean
JADE	4.1	4.7	4.7	4.5	4.5
JADE-mut	3.8	4.9	4.7	5.3	4.7
jDE-mut	5.5	5.5	5.4	4.9	5.3
jDE	5.6	5.7	5.6	4.9	5.5
EPSDE	7.1	5.7	5.5	6	6.1
EPSDE-mut	7.6	5.7	5.6	6.2	6.3
SaDE-mut	5.3	7.6	6.7	6.7	6.6
SaDE	5.5	7.3	7.1	6.7	6.6
CoDE-mut	8.9	6.7	7.4	7.5	7.6
CoDE	9	7.4	7.5	7.5	7.8
DE-mut	7.7	8.4	8.1	8.6	8.2
DE	7.9	8.4	9.7	9.2	8.8

using Friedman test for medians of function-error values. The null hypothesis on the equal performance of the algorithms was rejected, the achieved p value for rejection was $p < 5 \times 10^{-7}$. Mean ranks of the algorithms are presented in Table 2. Note that the algorithm winning uniquely in all the problems has the mean rank 1 and another algorithm being a unique looser in all the problems has the mean rank 12. In the last column of Table 2, the average mean rank is computed for all dimensions. It is

obvious that the least mean rank is for the original JADE variant. Interesting is the fact that all pairs of the original DE and the corresponding counterpart are together. Such a newly proposed DE variant outperforms the original algorithm in four out of six cases.

In some problems with many local-solution areas, DE algorithm often gets stuck. In these situations, the individuals of P are located in very small (local-solution) area and standard DE algorithm is not able to jumped-out. When the individual from the proposed auxiliary R population is in a mutation occasionally applied, it promises to use individuals out of the local solution area. On the other side, provided results show in some problems, that using the auxiliary memory decreases the convergence of DE. It could be caused by the fact that pr is set to fixed value. Some adaptive mechanism for pr value could be helpful.

The value of the fundamental control parameter pr influences the efficiency of algorithm. For smaller pr values, very small auxiliary R population of initialized individuals is used. This means that probability to select some individual of R in the mutation is very small and moving the population from local solution area is rarely. On the other side, when pr is set to big value (i.e. close to 1), the initialized points from R are applied more frequently (almost in each mutation) and the convergence of the algorithm is decreased.

Conclusions

The experimental comparison showed that the newly proposed enhanced mutation variant increased the performance of five state-of-the-art DE variants and also of the standard DE variant in some of the test problems. The number of wins of the new variants was smaller in the case of very efficiency JADE (30 %) and EPSDE (33 %) variants. The best performance was detected in the standard DE/rand/1/bin (62 %), jDE (48 %) or CoDE (47 %) variants.

Based on the Friedman rank test for median values, we can see that the best mean rank was acquired by the original JADE variant. No one DE variant has the best or the worst results for all problems and dimension levels. This fact only validates the No-Free-Lunch theorem [15].

This first study of the proposed mutation method showed that only several initialized individuals kept during the search process could significantly increase the performance of DE. There are many possible parameters to study and improve proposed technique. The study of the control parameters of the auxiliary population remains a challenge for further research.

Acknowledgement

This work was supported by the University of Ostrava from the project SGS08/UVAFM/2016.

Table 3
Medians of function values from 51 runs for DE/rand1/bin and DE/rand1/bin-mut variants and the results of Wilcoxon signed rank test for $D = 10, 30$

	$D = 10$			$D = 30$		
F	DE	DE-mut	Sign.	DE	DE-mut	Sign.
1	6.27E-02	**7.95E-07**	+	2.52E+06	**61214.9**	+
2	0	0	=	171.082	**7.86E-06**	+
3	20.3169	**20.2226**	+	20.9627	**20.5971**	+
4	20.3463	**9.90912**	+	204.622	**45.0517**	+
5	982.36	**411.36**	+	6857.85	**3978.27**	+
6	**0.41629**	1.61939	-	**3789.85**	6690.13	-
7	**0.14615**	0.19463	≈	5.49817	**3.02143**	+
8	**0.32203**	0.468	-	1103.15	**946.748**	≈
9	100.019	**100.018**	≈	**106.192**	106.686	-
10	143.109	**143.108**	≈	**693.532**	735.922	≈
11	**4.31265**	4.35499	≈	**410.239**	459.271	-
12	112.403	**112.239**	≈	112.553	**109.451**	+
13	**0.09273**	0.09715	-	**0.01032**	0.01068	-
14	6677.01	6677.01	≈	**42626.4**	43511.7	-
15	100	100	≈	100	100	≈

Table 4
Medians of function values from 51 runs for DE/rand1/bin and DE/rand1/bin-mut variants and the results of Wilcoxon signed rank test for $D = 50, 100$

	$D = 50$			$D = 100$		
F	DE	DE-mut	Sign.	DE	DE-mut	Sign.
1	4.79E+07	**348268**	+	6.21E+07	**1.34E+06**	+
2	121.174	**3.58E-03**	+	369.562	**36.8171**	+
3	21.1391	**20.8247**	+	21.3243	**21.2112**	+
4	403.504	**87.5563**	+	910.326	**226.249**	+
5	13041.1	**8637.04**	+	30157.3	**25113.5**	+
6	158580	**78616**	+	1.93E+07	**305807**	+
7	42.5321	**41.8693**	≈	137.958	**127.193**	≈
8	25810.7	**20548**	+	3.21E+06	**127927**	+
9	103.113	**102.228**	+	110.593	**108.105**	+
10	3792.33	**1266.27**	+	10286.9	**4194.43**	+
11	**442.227**	709.328	-	**760.799**	1779.12	-
12	201.536	**117.477**	+	200.406	**116.659**	+
13	**0.02498**	0.02607	-	**0.06267**	0.06543	-
14	**52669.4**	52682	-	**108853**	108887	-
15	100	100	≈	**100**	104.402	-

Table 5
Medians of function values from 51 runs for CoDE and CoDE-mut variants and the results of Wilcoxon signed rank test for $D = 10, 30$

	$D = 10$			$D = 30$		
F	CoDE	CoDE-mut	Sign.	CoDE	CoDE-mut	Sign.
1	0	0	≈	**4103.29**	8799.17	-
2	0	0	≈	0	0	≈
3	20.1112	**20.0963**	+	20.6215	**20.5205**	+
4	10.8867	**10.3145**	+	111.928	**101.869**	+
5	443.423	**431.602**	≈	4562.86	**4120.39**	+
6	27.4738	**25.4821**	≈	**758.52**	778.078	≈
7	1.11645	**0.94282**	+	8.44592	**8.41286**	≈
8	**2.32**	2.39408	≈	265.22	**251.065**	≈
9	**100.474**	100.596	-	**107.496**	107.712	-
10	149.322	**147.102**	+	**535.597**	537.875	≈
11	**3.98753**	4.10085	≈	410.262	**301.232**	+
12	**112.796**	112.996	-	110.897	**110.783**	≈
13	**0.09345**	0.09356	≈	**0.0111**	0.01115	≈
14	6662.87	6662.87	≈	42854.9	**42838.2**	≈
15	100	100	≈	100	100	≈

Table 6
Medians of function values from 51 runs for CoDE and CoDE-mut variants and the results of Wilcoxon signed rank test for $D = 50, 100$

	$D = 50$			$D = 100$		
F	CoDE	CoDE-mut	Sign.	CoDE	CoDE-mut	Sign.
1	**277480**	362414	-	1.36E+06	**1.27E+06**	≈
2	**1.96E-08**	1.02E-06	-	**1.67E-06**	2.51E-05	-
3	20.8712	**20.8052**	+	21.1895	**21.1579**	+
4	268.033	**250.965**	+	721.089	**686.159**	+
5	9764.8	**9185.22**	+	25291.2	**25060.5**	+
6	**4122.22**	4687.87	≈	316670	**296073**	≈
7	41.5762	**41.54**	≈	**135.926**	142.072	-
8	**1283.72**	1289.14	≈	**91204.4**	119871	-
9	**102.967**	103.066	-	110.522	**110.429**	≈
10	1877.67	**1732.36**	+	**2946.49**	3029.96	≈
11	**403.616**	417.833	-	**917.103**	1049.05	-
12	117.295	**117.154**	≈	**115.211**	115.435	≈
13	**0.0284**	0.02858	-	0.07484	**0.0704**	+
14	52680.6	52680.6	≈	108891	**108885**	≈
15	100	100	≈	100	100	≈

Table 7
Medians of function values from 51 runs for EPSDE and EPSDE-mut variants and the results of Wilcoxon signed rank test for $D = 10, 30$

	$D = 10$			$D = 30$		
F	EPSDE	EPSDE-mut	Sign.	EPSDE	EPSDE-mut	Sign.
1	0	0	≈	**1125.5**	2224.95	≈
2	0	0	≈	0	0	≈
3	20.1312	**20.1168**	≈	**20.6018**	20.6295	≈
4	11.4291	**11.2616**	≈	124.249	**123.511**	≈
5	**467.604**	495.116	≈	**4885.59**	4989.62	≈
6	**15.3399**	19.4987	≈	**941.091**	990.867	≈
7	0.4456	**0.44256**	≈	**7.08087**	7.22593	≈
8	**0.64177**	0.77123	≈	**333.416**	391.655	≈
9	**100.002**	100.003	-	105.659	**105.444**	≈
10	**141.556**	141.67	≈	**544.848**	551.536	≈
11	3.33224	**3.31824**	≈	404.124	404.124	≈
12	**112.07**	112.185	≈	110.261	**110.226**	≈
13	0.09273	0.09273	≈	0.01036	**0.01032**	≈
14	**6670.66**	6677.01	≈	42793	**42784.3**	≈
15	100	100	≈	100	100	≈

Table 8
Medians of function values from 51 runs for EPSDE and EPSDE-mut variants and the results of Wilcoxon signed rank test for $D = 50, 100$

	$D = 50$			$D = 100$		
F	EPSDE	EPSDE-mut	Sign.	EPSDE	EPSDE-mut	Sign.
1	**175045**	177178	≈	**587919**	609679	≈
2	0	0	≈	0	0	≈
3	20.8877	**20.8681**	≈	**21.2029**	21.2111	≈
4	**286.909**	288.666	≈	**779.656**	779.749	≈
5	10383.8	**10305.2**	≈	27304.9	**27242.8**	≈
6	**2172.48**	2702.65	≈	203293	**175716**	≈
7	**41.0566**	41.1552	≈	136.885	**135.937**	≈
8	**869.352**	1086.5	≈	**35186**	39968.2	≈
9	**102.456**	102.502	≈	**108.614**	108.674	≈
10	1003.44	**938.6**	≈	**3020.22**	3281.25	≈
11	439.034	**437.629**	≈	**854.065**	862.279	≈
12	201.536	**117.442**	+	200.406	**117.141**	≈
13	0.02516	**0.02502**	≈	0.06269	**0.06239**	≈
14	**52662.7**	52678.2	≈	**108865**	108870	≈
15	100	100	≈	100	100	≈

Table 9
Medians of function values from 51 runs for JADE and JADE-mut variants and the results of Wilcoxon signed rank test for $D = 10, 30$

	$D = 10$			$D = 30$		
F	JADE	JADE-mut	Sign.	JADE	JADE-mut	Sign.
1	0	0	≈	**1.31028**	4.04967	-
2	0	0	≈	0	0	≈
3	20.0579	**20.052**	≈	**20.2779**	20.2812	≈
4	3.55754	**3.48658**	≈	26.5575	**25.3809**	≈
5	**52.7562**	57.1781	≈	**1699.69**	1734.4	≈
6	0.41629	**0.41629**	≈	**941.019**	1019.68	≈
7	**0.30991**	0.31690	≈	7.87472	**7.66545**	+
8	**0.53595**	0.57577	≈	**219.249**	251.085	≈
9	100	100	≈	106.547	**106.395**	≈
10	143.108	143.108	≈	**715.256**	742.593	≈
11	**2.97343**	3.02492	≈	**408.796**	408.851	≈
12	111.798	**111.767**	≈	**108.972**	109.142	≈
13	0.09273	**0.09273**	≈	0.01049	**0.01041**	≈
14	6670.66	6670.66	≈	43620	**43410.6**	≈
15	100	100	≈	100	100	≈

Table 10
Medians of function values from 51 runs for JADE and JADE-mut variants and the results of Wilcoxon signed rank test for $D = 50, 100$

	$D = 50$			$D = 100$		
F	JADE	JADE-mut	Sign.	JADE	JADE-mut	Sign.
1	7272.18	**5756.84**	≈	**76376.1**	86061.3	≈
2	0	0	≈	0	0	≈
3	**20.3602**	20.3676	≈	**20.4627**	20.4655	≈
4	**51.7394**	57.042	-	**154.44**	157.685	≈
5	3532.94	**3486.93**	≈	**10313.6**	10331	≈
6	2661.89	**2537.13**	≈	**11887.2**	13762	≈
7	42.5838	**42.2447**	≈	**116.745**	127.462	≈
8	**1197.8**	1215.5	≈	**3835.97**	3945.31	≈
9	**102.66**	102.765	≈	**110.555**	111.139	-
10	1760.11	**1572.17**	+	**4361.36**	4597.21	≈
11	522.613	**513.155**	≈	**1259.63**	1298.47	≈
12	**116.342**	116.451	≈	116.311	**115.426**	≈
13	**0.0255**	0.02554	≈	0.0633	**0.06324**	≈
14	**52678.2**	52680.6	≈	**108881**	108887	≈
15	100	100	≈	**101.302**	101.463	≈

Table 11
Medians of function values from 51 runs for jDE and jDE-mut variants and the results of Wilcoxon signed rank test for $D = 10, 30$

	$D = 10$			$D = 30$		
F	jDE	jDE-mut	Sign.	jDE	jDE-mut	Sign.
1	**0**	5.88E-08	-	34246.4	**31656.3**	≈
2	0	0	≈	0	0	≈
3	20.0689	**20.0629**	≈	**20.3096**	20.3108	≈
4	5.45472	**5.39326**	≈	**39.9106**	40.8473	≈
5	231.063	**195.111**	≈	2428.06	**2422.21**	≈
6	9.86734	**9.59008**	≈	**1018.45**	1550.9	-
7	0.36721	0.36984	≈	8.56411	**8.40703**	≈
8	0.47653	0.44016	≈	219.256	**199.384**	≈
9	100.007	100.01	≈	106.102	**106.072**	≈
10	141.618	141.548	≈	637.044	**635.476**	≈
11	3.19005	3.18598	≈	**404.124**	408.696	≈
12	112.458	112.554	≈	109.716	**109.524**	+
13	0.09273	0.09273	≈	**0.01033**	0.01035	≈
14	6662.87	6662.87	≈	43419.2	**43410.6**	≈
15	100	100	≈	100	100	≈

Table 12
Medians of function values from 51 runs for jDE and jDE-mut variants and the results of Wilcoxon signed rank test for $D = 50, 100$

	$D = 50$			$D = 100$		
F	jDE	jDE-mut	Sign.	jDE	jDE-mut	Sign.
1	396624	**374644**	≈	1.43E+06	**1.42E+06**	≈
2	0	0	≈	0	0	≈
3	20.4252	**20.4199**	≈	20.6601	**20.6526**	≈
4	83.9694	**83.924**	≈	225.627	**199.583**	+
5	**4638.29**	4641.82	≈	12710.9	**12530.1**	≈
6	**30558.7**	38689.4	≈	**188422**	226244	-
7	**43.0974**	44.4463	≈	**134.681**	135.603	-
8	**8003.51**	9033.64	≈	79939.2	**76601.8**	≈
9	**102.194**	102.199	≈	**107.077**	107.225	≈
10	1428.24	**1325.64**	≈	**3791.2**	3863.28	≈
11	488.976	**467.289**	≈	1079.89	**946.754**	≈
12	116.835	**116.831**	≈	114.994	**114.9**	≈
13	0.02497	**0.02491**	≈	**0.06174**	0.0619	≈
14	52661	52661	≈	108887	108887	≈
15	100	100	≈	100	100	≈

Table 13
Medians of function values from 51 runs for SaDE and SaDE-mut variants and the results of Wilcoxon signed rank test for $D = 10, 30$

	$D = 10$			$D = 30$		
F	SaDE	SaDE-mut	Sign.	SaDE	SaDE-mut	Sign.
1	0	0	≈	**7216.46**	7438.64	≈
2	0	0	≈	0	0	≈
3	**20.0823**	20.0842	≈	**20.385**	20.3967	≈
4	4.83555	**4.44725**	≈	34.7638	**33.6348**	≈
5	180.634	**161.037**	≈	**2674.22**	2725.79	≈
6	**4.68252**	6.05311	≈	**1255.06**	1457.19	≈
7	0.38712	**0.33234**	≈	8.19227	**8.09298**	≈
8	1.26452	**1.11676**	+	**302.761**	410.456	≈
9	100	100	≈	106.853	**106.519**	≈
10	**141.655**	143.109	≈	814.879	**798.48**	≈
11	**3.16052**	3.18455	≈	**417.978**	426.424	≈
12	111.984	**111.979**	≈	**109.505**	109.614	≈
13	0.09273	0.09273	≈	0.01044	**0.01041**	≈
14	6677.01	**6670.66**	≈	**43610.6**	43806.1	≈
15	100	100	≈	100	100	≈

Table 14
Medians of function values from 51 runs for SaDE and SaDE-mut variants and the results of Wilcoxon signed rank test for $D = 50, 100$

	$D = 50$			$D = 100$		
F	SaDE	SaDE-mut	Sign.	SaDE	SaDE-mut	Sign.
1	**86925.6**	89696.9	≈	**419702**	513455	-
2	0	0	≈	1.88E-06	**2.22E-07**	+
3	**20.5758**	20.577	≈	**20.8954**	20.9055	-
4	92.14	**80.3453**	+	262.669	**190.037**	+
5	6012.55	**5785.05**	≈	17364.8	**17195.9**	+
6	**15052**	16486.5	≈	**84926.9**	87087.2	-
7	49.6038	**49.5846**	≈	**113.103**	142.63	≈
8	**3334.2**	3442.65	≈	**26663**	37648.8	-
9	102.431	**102.334**	+	108.714	**108.285**	+
10	**1734.43**	1879.22	≈	**3801.22**	4080.34	-
11	681.752	**655.88**	+	1860.02	**1765.77**	+
12	116.769	**116.624**	≈	115.608	**115.312**	+
13	0.02553	**0.02545**	≈	0.06383	**0.06324**	+
14	52698	**52693.6**	≈	108912	**108895**	+
15	100	100	≈	**101.902**	101.976	-

References

[1] J. Brest, S. Greiner, B. Boškovič, M. Mernik and V. Žumer: Self-adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems, *IEEE Transactions on Evolutionary Computation*, vol. 10, pp. 646–657, 2006.

[2] P. Bujok and J, Tvrdík: A Comparison of Various Strategies in Differential Evolution, In: *R. Matoušek (Ed.) MENDEL 2011 - 17th International Conference On Soft Computing, Brno, Czech Republic*, pp. 48–55, 2011.

[3] S. Das and P. N. Suganthan: Differential evolution: A survey of the state-of-the-art, *IEEE Transactions on Evolutionary Computation*, vol. 15, pp. 27–54, 2011.

[4] S. Das, S. S. Mullick, P. N. Suganthan: Recent advances in differential evolution An updated survey, *Swarm and Evolutionary Computation*, Vol. 27, pp. 1–30, 2016.

[5] J. J. Liang, P. N. Suganthan, and Q. Chen: Problem definitions and evaluation criteria for the CEC 2015 competition on learning-based real-parameter single objective optimization, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Nanyang Technological University, Tech. Rep., 2014.

[6] R. Mallipeddi, P. N. Suganthan, Q. K. Pan and M. F. Tasgetiren: Differential evolution algorithm with ensemble of parameters and mutation strategies, *Applied Soft Computing*, vol. 11, pp. 1679–1696, 2011.

[7] F. Neri and V. Tirronen: Recent advances in differential evolution: a survey and experimental analysis, *Artificial Intelligence Review*, vol. 33, pp. 61–106, 2010.

[8] A. P. Piotrowski: Review of Differential Evolution population size, *Swarm and Evolutionary Computation*, Vol. 32, pp. 1–24, 2017.

[9] W. Qian and A. Li: Adaptive differential evolution algorithm for multiobjective optimization problems, *Applied Mathematics and Computation*, vol. 201, no. 12, pp. 431–440, 2008.

[10] A. K. Qin, V. L. Huang, and P. N. Suganthan: Differential evolution algorithm with strategy adaptation for global numerical optimization, *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 398-417, 2009.

[11] R. Storn and K. V. Price: Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces, *Journal of Global Optimization*, vol. 11, pp. 341–359, 1997.

[12] Y. Wang, Z. Cai, Q. Zhang: Differential evolution with composite trial vector generation strategies and control parameters. *IEEE Transactions on Evolutionary Computation*, vol. 15, pp. 55-66, 2011.

[13] Y. Wang, Z.-Z. Liu, J. Li, H.-X. Li, G. G. Yen: Utilizing cumulative population distribution information in differential evolution, *Applied Soft Computing*, Vol. 48, pp. 329–346, 2016.

[14] Y. Wang, Z.-Z. Liu, J. Li, H. X. Li and J. Wang: On the selection of solutions for mutation in differential evolution. *Frontiers of Computer Science*. In press, DOI: 10.1007/s11704-016-5353-5.

[15] D. H. Wolpert and W. G. Macready: No free lunch theorems for optimization, *IEEE Transactions on Evolutionary Computation*, vol. 1, pp. 67–82, 1997.

[16] Z. Yang, K. Tang, and X. Yao: Self-adaptive differential evolution with neighborhood search, *IEEE Congress on Evolutionary Computation, CEC 2008*, pp. 1110–1116, 2008.

[17] A. Zamuda and J. Brest: Self-adaptive control parameters? randomization frequency and propagations in differential evolution, *Swarm and Evolutionary Computation*, vol. 25, pp. 72–99, 2015.

[18] J. Zhang and A. C. Sanderson: JADE: Adaptive differential evolution with optional external archive, *IEEE Transactions on Evolutionary Computation*, vol. 13, pp. 945–958, 2009.