2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

Controlled Restart in Differential Evolution Applied to CEC2014 Benchmark Functions

Radka Poláková
Centre of Excellence IT4Innovations,
Institute for Research and
Applications of Fuzzy Modeling,
University of Ostrava,
30. dubna 22, 701 03 Ostrava,
Czech Republic
Email: radka.polakova@osu.cz

Josef Tvrdík
Department of Computer Science,
Centre of Excellence IT4Innovations,
Institute for Research and
Applications of Fuzzy Modeling,
University of Ostrava,
30. dubna 22, 701 03 Ostrava,
Czech Republic
Email: josef.tvrdik@osu.cz

Petr Bujok
Department of Computer Science,
University of Ostrava,
30. dubna 22, 701 03 Ostrava,
Czech Republic
Email: petr.bujok@osu.cz

***Abstract*—A controlled restart in differential evolution (DE) is proposed. The conditions of restart are derived from the difference of maximum and minimum values of the objective function and the estimated maximum distance among the points in the current population. The restart is applied in a competitive-adaptation variant of DE. This DE algorithm with the controlled restart is used in the solution of the benchmark problems defined for the CEC 2014 competition. Two control parameters of restart are set up intuitively. The population size, which is the only control parameter of competitive-adaptation variant of DE, is set up to the values based on a short preliminary experimentation.**

I. Introduction

Differential evolution (DE) was proposed by Storn and Price [1], [2] as a simple heuristic for solving continuous single-objective optimization problems with a real-value objective function. The search space (domain) Ω is specified by lower (a_j) and upper (b_j) bounds of each component j, $\Omega = \prod_{j=1}^{D}[a_j, b_j],\ a_j < b_j,\ j = 1, 2, \ldots, D$, D is the dimension of the search space. The global minimum point $\boldsymbol{x}^*$ satisfying the condition $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x})$ for $\forall \boldsymbol{x} \in \Omega$ is the solution of the problem.

DE algorithm has become one of the most frequently evolutionary algorithms applied to the solution of the continuous global optimization problems in recent years [3]. The algorithm has been also intensively studied and many modifications have appeared in literature. Recent results of research in DE are comprehensively summarized by Neri and Tirronen [4] and by Das and Suganthan [5].

The remaining part of the paper is organized as follows: DE algorithm is described in Section II. Mechanism of controlled restart is proposed in Section III. Section IV presents the adaptive mechanism used in competitive DE and the variant of competitive DE used in the experiments is also described in this section. Experimental setting is specified in Section V. The results of experiments are presented in Section VI including the time complexity of the algorithm. Conclusions are drawn in Section VII.

II. DE Algorithm

The DE algorithm works with a population of individuals (*NP* points in domain Ω) that are considered as candidates of solution. A parameter *NP* is called the size of the population. The population develops iteratively by using evolutionary operators of selection, mutation, and crossover generation by generation. Let us denote two subsequent generations by P and Q. Applications of evolutionary operators in the old generation P create individuals for a new generation Q. After completing the new generation Q, the generation Q becomes the old generation P for next iteration. The DE algorithm written in pseudo-code is shown in Algorithm 1.

Algorithm 1 Differential evolution

```
1:  generate an initial population P = (x_1, x_2, ..., x_NP),
    x_i ∈ Ω distributed uniformly
2:  evaluate f(x_i), i = 1, 2, ..., NP
3:  while stopping condition not reached do
4:    for i = 1 to NP do
5:      generate a trial vector y
6:      evaluate f(y)
7:      if f(y) ≤ f(x_i) then
8:        insert y into new generation Q
9:      else
10:       insert x_i into new generation Q
11:     end if
12:   end for
13:   P := Q
14: end while
```

A new trial point $\boldsymbol{y}$ (line 5 in Algorithm 1) is generated by using mutation and crossover. There are various types of mutation and crossover. The most popular mutation type proposed in [1] and called DE/rand/1 generates the mutant point $\boldsymbol{u}$ by adding the weighted difference of two points,

$$\boldsymbol{u} = \boldsymbol{r}_1 + F\left(\boldsymbol{r}_2 - \boldsymbol{r}_3\right), \quad F > 0\,, \tag{1}$$

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

Kaelo and Ali [6] proposed a slight modification of mutation defined in (1):

$$\boldsymbol{u} = \dot{\boldsymbol{r}}_1 + F\left(\dot{\boldsymbol{r}}_2 - \dot{\boldsymbol{r}}_3\right), \tag{2}$$

where $\dot{\boldsymbol{r}}_1$ is the tournament best among $\boldsymbol{r}_1$, $\boldsymbol{r}_2$, and $\boldsymbol{r}_3$, i.e. $\dot{\boldsymbol{r}}_1 = \arg\min_{i\in\{1,2,3\}} f(\boldsymbol{r}_i)$, where $\boldsymbol{r}_1, \boldsymbol{r}_2$, and $\boldsymbol{r}_3$ are three mutually distinct points randomly taken from population P, not coinciding with the current point $\boldsymbol{x}_i$, like in the case of the DE/rand/1. The points $\dot{\boldsymbol{r}}_2$ and $\dot{\boldsymbol{r}}_3$ are the remaining points of the three randomly selected points. This type of mutation is denoted DE/randrl/1 hereafter. Based on the experimental results in [6], it was found that this DE/randrl/1 mutation can increase the speed of the search by almost 30 % without decreasing the reliability of the search when compared to the DE/rand/1 mutation.

The elements y_j, $j = 1, 2, \ldots, D$, of the trial point $\boldsymbol{y}$ are built up by the crossover of the current point $\boldsymbol{x}_i$ and the mutant point $\boldsymbol{u}$. The most frequently used kind of crossover is called *binomial*. It uses the following rule of combination of parents' elements

$$y_j = \begin{cases} u_j & \text{if} \quad U_j \le \mathit{CR} \quad \text{or} \quad j = l \\ x_{ij} & \text{if} \quad U_j > \mathit{CR} \quad \text{and} \quad j \ne l\,, \end{cases} \tag{3}$$

where l is a randomly chosen integer from $\{1, 2, \ldots, D\}$, $U_1, U_2, \ldots, U_D$ are independent random variables uniformly distributed in $[0,\ 1)$, and $\mathit{CR} \in [0, 1]$ is an input parameter influencing the number of elements to be exchanged by crossover. The rule given by Eq. (3) ensures that at least one element of vector $\boldsymbol{x}_i$ is changed, even if $\mathit{CR} = 0$. The DE variants applying binomial crossover according to (3) are denoted DE/$\cdot$/$\cdot$/bin in literature.

The *exponential* crossover usually denoted DE/$\cdot$/$\cdot$/exp was also proposed by Price and Storn [1]. The exponential crossover in DE is similar to the two-point crossover in genetic algorithms. For exponential crossover, the starting position of crossover ($k = 1$) is randomly chosen from $\{1, 2, \ldots, D\}$, and L consecutive elements (counted in circular manner) are taken from the mutant vector $\boldsymbol{u}$. Probability of replacing the k-th element in the sequence $1, 2, \ldots, L, \quad L \le D$ decreases exponentially with increasing k.

Let us consider the probability p_m of replacing the element of $\boldsymbol{x}_i$ by the element of the mutant vector $\boldsymbol{u}$. The mean value of replaced elements is then $p_m \times D$. The relation between the p_m and control parameter *CR* was studied in detail by Zaharie [7]. For binomial crossover, the relation between p_m and control parameter *CR* is linear,

$$p_m = \mathit{CR}(1 - 1/D) + 1/D, \tag{4}$$

while for exponential crossover the relationship is strongly non-linear and the difference from linearity increases with increasing dimension of the problem. Zaharie's result for exponential crossover can be rewritten in the form of polynom equation

$$\mathit{CR}^D - \ D\ p_m\ \mathit{CR} + \ D\ p_m - 1 = 0. \tag{5}$$

The crossover parameter *CR* satisfies the conditions: $\mathit{CR} = 0$ for $p_m = 1/D$ and $\mathit{CR} = 1$ for $p_m = 1$. It is apparent that the equation (5) has the only one real root in the open interval of $(0, 1)$ for $p_m \in (1/D, 1)$. Thus, for given p_m we are able to find unique corresponding value of *CR* from (5). This is applied in the implementation of competitive DE variant used in this study.

Differential evolution has a few control parameters only, namely the size of population *NP*, mutation strategy, crossover type, and couple of parameters F and *CR*. However, the efficiency of differential evolution is very sensitive especially regarding the setting of F and *CR* values. The most suitable control-parameter values for a specific problem may be found by trial-and-error tuning, which requires a lot of time. There are some recommendations for setting of these parameters but they are not applicable for all the optimization problems. Due to these facts, several new adaptive variants of DE have been recently proposed, some of them [8]–[11] are considered as the state-of-the-art adaptive variants of DE. However, the research in the adaptive mechanisms of DE is intensive and many new variants have appeared recently, e.g. [12]–[20].

III. Controlled Restart

DE usually converges quickly to the global minimum, especially in easier optimization problems (separable, unimodal etc.). However, it is also known that there are optimization problems when the search at some stage is not able to amend the population. The search process stagnates and next continuation of the search is wasting of computational time. This phenomenon was described and partially explained by Zelinka and Lampinen in [21]. Such behavior of DE was also observed in some CEC 2013 test problems [22], where a minimum function value remained the same for thousands of generations until the search stopped. It is apparent that the population in such situation needs a new impuls. Two questions arise in this context:

- What kind of the impuls could be useful?
- How to detect this type of stagnation?

An attempt to solve these problems was done in [23], where the population was enlarged by new points evolved using information from previous generations. However, it was found in our preliminary research that the change of a part of the population is not a sufficient impuls to interrupt the stagnation, probably due to the fact that new points with higher values of the objective function are discarded quickly from the current population and their effect is not strong enough to leave the attraction region found before. It indicates that a completion of new initialization of the population (restart of the search) is needed to enable finding another region with small function values. When considering the conditions for a restart and how to detect them automatically, we can infer from possible reasons of the stagnation: It can be caused by small diversity of population and/or convergence to a local minimum where the population is trapped.

We denote f_{min} and f_{max} the minimum and maximum function values in the current population, respectively. A very small difference $f_{max} - f_{min}$ can indicate a trapping at a local minimum. To recognize a small variability of the current population is not so easy. The variability of the population can be measured by a maximum possible distance of points in the current population defined as follows:

Let $\boldsymbol{x}_{min}$ be the vector of minimum values of coordinates and let $\boldsymbol{x}_{max}$ be the vector of maximum values of coordinates in the current population. Then the upper limit of the maximum possible Euclidian distance of points in the current population can be evaluated by

$$\textit{maxdist} = \sqrt{(\boldsymbol{x}_{max} - \boldsymbol{x}_{min})' \cdot (\boldsymbol{x}_{max} - \boldsymbol{x}_{min})}. \quad (6)$$

Then the condition for restart can be formed as

$$f_{max} - f_{min} < \varepsilon_f \quad \text{AND} \quad \textit{maxdist} < \varepsilon_d, \quad (7)$$

where ε_f and ε_d are input parameters controlling restart.

Before each restart the solution found is stored in order to be compared with the solution found later (or before) and the best solution of all is the solution found by the algorithm with the controlled restart.

IV. Competitive Differential Evolution

Competitive setting of the DE strategies and the control parameters were originally proposed in [24]. In this self-adaptive approach, we choose randomly among H different DE strategies or settings of control parameters (F, ***CR***) with probabilities $q_h,\ h = 1, 2, \ldots, H$.

These probabilities change according to the success rate of the settings in preceding steps of the search process. The hth setting is considered successful if it generates a trial point $\boldsymbol{y}$ better than its counter-partner $\boldsymbol{x}_i$ in the old generation P, i.e. $f(\boldsymbol{y}) \leq f(\boldsymbol{x}_i)$, see line 7 of Algorithm 1. Probability q_h is evaluated as the relative frequency

$$q_h = \frac{n_h + n_0}{\sum_{j=1}^{H}(n_j + n_0)}\ , \quad (8)$$

where n_h is the current count of the hth setting's successes, and $n_0 > 0$ is a constant. The input parameter $n_0 > 1$ prevents a dramatic change in q_h by one random successful use of the hth parameter setting. To avoid degeneration of the search process, the current values of q_h are reset to their starting values $q_h = 1/H$ if any probability q_h decreases below the given limit $\delta > 0$ during the search process. The adaptive variant of DE is called competitive differential evolution (CDE) hereafter.

Several settings of CDE were compared in different benchmark tests [25], [26]. The CDE variant denoted *b6e6rl* was found as one of the most efficient among all the tested CDE variants. The variant uses twelve different DE strategies or parameter settings. This version of CDE was also compared with the state-of-the-art adaptive DE algorithms [8]–[11] and two versions of DE algorithm with composite trial vector generation strategies and control parameters [12]. The *b6e6rl* variant of CDE appeared the most reliable and the second fastest algorithm among those seven algorithms on the benchmark set of six shifted functions at three levels of dimension ($D = 10$, 30, and 100) [27].

The *b6e6rl* variant of CDE uses two DE strategies, namely DE/randrl/1/bin and DE/randrl/1/exp, each with six different settings of $(F, \textit{CR})$. The strategies and settings are listed in Table I. The values of p_1, p_2, and p_3 are set up equidistantly in the open interval of $(1/D, 1)$. The value of p_2 is in the middle of $(1/D, 1)$, p_1 is in the middle of $(1/D, p_2)$, and p_3 in the middle of $(p_2, 1)$. The values of p_i, $i = 1, 2, 3$, are set up automatically with respect to the dimension of the problem at the start of the algorithm as well as the corresponding values of $\textit{CR}_i$ evaluated as roots of the polynom (5).

TABLE I. DE strategies and parameter settings competing in *b6e6rl* variant of CDE

h	mutation	crossover	F	CR	p_m
1	randrl/1	binomial	0.5	0	
2			0.5	0.5	
3			0.5	1	
4			0.8	0	
5			0.8	0.5	
6			0.8	1	
7	randrl/1	exponential	0.5	CR_1	p_1
8			0.5	CR_2	p_2
9			0.5	CR_3	p_3
10			0.8	CR_1	p_1
11			0.8	CR_2	p_2
12			0.8	CR_3	p_3

The values of mutation probability and the corresponding values of ***CR*** applied to the problems of $D = 10$, 30, 50 and 100 are shown in Table II. Mutation according to (2) can cause that a new trial point $\boldsymbol{y}$ moves out of the domain Ω, which means the violation of the boundary constraints. In such a case, the values of $y_j \notin [a_j, b_j]$ can be corrected by return into Ω using transformation $y_j \leftarrow 2 \times a_j - y_j$ or $y_j \leftarrow 2 \times b_j - y_j$ for the violated components. This technique is frequently used and also implemented in *b6e6rl* variant of CDE algorithm. The parameters controlling the competition of DE strategies and control-parameter settings are in *b6e6rl* set up to the recommended values succeeded in previous applications of the algorithm [27], i.e. $n_0 = 2$ and $\delta = 1/(H * 5) = 0.0167$.

TABLE II. Values of mutation probability and the corresponding values of ***CR*** for exponential crossover

	$D = 10$		$D = 30$	
i	p_i	$\textit{CR}_i$	p_i	$\textit{CR}_i$
1	0.3250	0.7011	0.2750	0.8815
2	0.5500	0.8571	0.5167	0.9488
3	0.7750	0.9418	0.7583	0.9801
	$D = 50$		$D = 100$	
i	p_i	$\textit{CR}_i$	p_i	$\textit{CR}_i$
1	0.2650	0.9262	0.2525	0.9611
2	0.5100	0.9688	0.4950	0.9837
3	0.7550	0.9880	0.7475	0.9938

V. Experimental Setting

All computations were carried out on a standard PC with Windows 8, Intel(R) Core(TM)i5-323OM CPU, 2.60GH, 2.60GHz, 8 GB RAM.

The algorithm *b6e6rl* with controlled restart described above is implemented in Matlab 2010a and this environment was used for experiments. Experimental setting follows the requirements given in the report [28]. 30 minimization problems are defined in the report, the source code of functions in C cec14_func.cpp from December 20, 2013 was downloaded

TABLE III. VALUES OF FUNCTION ERRORS FOR *b6e6rl* WITH CONTROLLED RESTART, $D = 10$

F	Best	Worst	Median	Mean	Std
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	34.780	0	1.125	4.979
5	2.78E-08	20.094	20.052	18.453	4.423
6	0	0	0	0	0
7	0	0.047	0.015	0.017	0.013
8	0	0	0	0	0
9	2.464	7.878	4.850	4.895	1.076
10	0	0.062	0	0.001	0.009
11	35.860	330.908	198.482	196.524	79.947
12	0.193	0.452	0.281	0.293	0.056
13	0.037	0.191	0.133	0.128	0.033
14	0.032	0.164	0.111	0.111	0.029
15	0.486	1.145	0.839	0.832	0.157
16	1.045	2.314	1.909	1.872	0.255
17	0	38.800	0.212	1.398	5.576
18	0.006	2.148	0.472	0.621	0.576
19	0.032	0.458	0.115	0.142	0.092
20	2.49E-05	0.422	0.012	0.056	0.083
21	1.08E-05	17.260	0.485	0.787	2.373
22	0.001	0.630	0.055	0.154	0.173
23	329.457	329.457	329.457	329.457	2.18E-10
24	107.915	117.504	112.240	112.212	2.070
25	100	200.590	118.080	120.866	14.863
26	100.037	100.189	100.118	100.117	0.033
27	1.129	400.154	2.107	61.608	124.098
28	356.827	458.317	357.103	363.388	14.726
29	100	223.249	221.759	217.788	20.565
30	462.307	534.998	462.782	467.274	13.998

from the web page [28] and compiled via mex command (mex cec14_func.cpp).

Search range for all the test functions is $[-100, 100]^D$. The tests were carried out at four levels of dimension, 51 repeated runs per the test function and the dimension of the problem. The random generator was set up by rand('state',sum(100*clock)) statement at the start of each run. The run was stopped if the prescribed MaxFES was reached, i.e. when MaxFES $= 1 \times 10^5$ for $D = 10$, MaxFES $= 3 \times 10^5$ for $D = 30$, MaxFES$= 5 \times 10^5$ for $D = 50$, and MaxFES$= 10^6$ for $D = 100$.

Due to adaptive features of the *b6e6rl* algorithm, the only control parameter needed to be set is the population size. It was found in previous experiments with adaptive DE variants [27] that for a good performance it is sufficient much smaller population size than $NP \simeq 10 \times D$ recommended for many evolutionary algorithms including standard DE [1]. The adaptive DE variants performed well with the population size from 20 to $5 \times D$ in the problems of low dimension and with $NP \leq 100$ in the problems of higher dimension. Thus, the population size *NP* was set up to $NP = 50$.

The parameters controlling the condition of the restart, see (7), were set up as follows:

- $\varepsilon_f = 1 \times 10^{-8}$
- $\varepsilon_d = 1$.

The value of ε_d is based on the ratio of measures of sets. The search space is a measure of $\mu(\Omega) = (\prod_{j=1}^{D}[b_j - a_j])$, which is 200^D in all the test problems. The measure of the subset of the search space containing all the points of the current population is $\mu(S)$ and can be approximated by $\mu(S) \approx \mathit{maxdist}^D$. The setting $\varepsilon_d = 1$ then means the ratio of the measures $\mu(S)/\mu(\Omega) < 1/200^D$.

TABLE IV. VALUES OF FUNCTION ERRORS FOR *b6e6rl* WITH CONTROLLED RESTART, $D = 30$

F	Best	Worst	Median	Mean	Std
1	5407.71	149937.4	30051.7	40537.3	34343.6
2	0	0	0	0	0
3	0	0	0	0	0
4	0.002	0.542	0.158	0.162	0.095
5	20.200	20.332	20.273	20.271	0.028
6	0.003	15.040	13.228	12.126	3.441
7	0	1.17E-08	0	2.30E-10	1.64E-09
8	0	0	0	0	0
9	33.254	60.687	42.687	44.102	6.223
10	0	0.083	0.042	0.035	0.025
11	1372.75	2579.87	1995.69	1980.10	220.159
12	0.235	0.421	0.346	0.339	0.047
13	0.207	0.444	0.347	0.336	0.052
14	0.192	0.305	0.241	0.243	0.025
15	4.157	7.046	5.919	5.808	0.720
16	8.701	10.285	9.603	9.539	0.364
17	188.182	9775.64	1505.39	2116.29	1926.73
18	10.479	90.045	24.043	28.702	15.730
19	3.156	8.686	4.185	4.386	0.979
20	3.944	40.377	13.262	15.379	6.534
21	3.089	667.589	207.816	243.108	143.770
22	14.649	271.065	28.549	63.615	66.370
23	315.244	315.244	315.244	315.244	1.20E-09
24	218.525	224.184	222.691	222.881	0.930
25	202.585	204.585	203.224	203.290	0.472
26	100.251	100.491	100.353	100.351	0.048
27	300	401.789	343.953	351.556	44.569
28	685.175	866.624	832.100	820.450	37.267
29	419.235	1113.22	767.443	786.838	112.297
30	485.722	4132.03	1193.05	1318.25	668.521

VI. RESULTS

The report [28] stated the obligatory content and structure of the experimental results, which is presented below.

TABLE V. COMPUTATIONAL COMPLEXITY

	T0	*T1* (s)	$\hat{T}2$ (s)	$(\hat{T}2 - T1)/T0$
$D = 10$	0.1421	1.52	14.42	90.78
$D = 30$	0.1421	1.72	15.35	95.92
$D = 50$	0.1421	2.13	17.12	105.49
$D = 100$	0.1421	3.90	20.08	113.86

A. Function errors

The values of function errors are shown in Tables III, IV, VI, and VII for $D = 10$, $D = 30$, $D = 50$, and $D = 100$, respectively. The tables are presented in the structure required by [28], the values of function error smaller than 1×10^{-8} were replaced by zeros.

TABLE VI. VALUES OF FUNCTION ERRORS FOR *b6e6rl* WITH CONTROLLED RESTART, $D = 50$

F	Best	Worst	Median	Mean	Std
1	137441	885694	333428	371051	179033
2	7.865	23746.7	2814.22	5011.63	5529.22
3	7.213	1639.67	126.845	238.587	343.228
4	0.685	98.103	19.516	49.464	34.620
5	20.319	20.408	20.367	20.366	0.022
6	22.648	31.612	27.336	27.489	2.127
7	1.12E-08	2.64E-08	1.72E-08	1.74E-08	3.63E-09
8	0	1.02E-08	0	2.00E-10	1.43E-09
9	78.270	126.365	100.697	102.090	10.450
10	0	0.087	0.037	0.035	0.018
11	3534.67	5004.79	4439.83	4395.23	314.704
12	0.260	0.424	0.354	0.349	0.033
13	0.317	0.561	0.480	0.469	0.049
14	0.208	0.759	0.274	0.287	0.075
15	9.757	14.959	12.801	12.626	1.180
16	16.218	18.634	17.784	17.766	0.445
17	3559.97	48665.8	15458.8	17529.4	10568.4
18	79.488	5506.14	670.759	1228.90	1312.963
19	9.477	77.125	11.906	13.302	9.228
20	89.248	2818.646	468.506	628.362	567.167
21	2569.74	149837	18104.7	24093.4	23442.16
22	33.013	1247.35	538.201	568.865	207.097
23	344.005	344.005	344.005	344.005	1.31E-09
24	255.608	267.776	263.986	262.030	3.633
25	205.089	210.290	206.853	207.073	1.251
26	100.288	200.081	100.498	118.058	38.338
27	359.385	1102.06	985.922	901.119	202.462
28	1028.12	1452.69	1236.12	1236.01	72.860
29	883.278	1612.83	1349.01	1317.36	202.812
30	8118.28	14199.3	9605.62	9604.60	913.833

TABLE VII. VALUES OF FUNCTION ERRORS FOR *b6e6rl* WITH CONTROLLED RESTART, $D = 100$

F	Best	Worst	Median	Mean	Std
1	701234	2715168	1194116	1307760	406951
2	10.246	103284	7882.61	22166.9	26930.3
3	5.734	4047.682	449.749	693.816	793.275
4	113.184	257.178	176.901	173.789	33.351
5	20.523	20.631	20.594	20.593	0.023
6	66.767	80.079	74.562	73.952	2.984
7	3.42E-08	8.31E-08	5.82E-08	5.68E-08	1.09E-08
8	0	1.78E-08	1.19E-08	1.08E-08	5.06E-09
9	243.954	384.144	322.467	320.655	34.358
10	0.012	0.354	0.050	0.053	0.045
11	10355.8	12671	11864.4	11808.6	480.471
12	0.421	0.568	0.508	0.504	0.032
13	0.405	0.649	0.530	0.523	0.052
14	0.180	0.253	0.219	0.218	0.016
15	31.828	53.597	41.182	40.866	4.470
16	37.213	40.794	39.832	39.701	0.654
17	70031.1	289755	187144	185701	53120.4
18	143.941	4745.07	715.966	940.903	865.470
19	55.457	129.367	94.495	94.720	12.022
20	2548.42	16968.48	6812.02	7725.58	3384.60
21	16485.7	205283.7	88171.4	89216.6	39231.3
22	1212.07	2307.73	1866.80	1881.67	230.866
23	348.235	348.235	348.235	348.235	2.24E-08
24	358.092	367.556	362.846	363.122	2.376
25	231.249	275.206	247.533	248.487	9.051
26	200.123	200.332	200.188	200.195	0.044
27	944.348	2263.27	2044.61	2024.92	190.436
28	2125.93	5237.90	2566.68	3052.01	767.930
29	1177.62	1929.66	1573.51	1605.85	180.403
30	5840.62	10883.6	8719.10	8535.46	1167.04

B. Algorithm complexity

Algorithm complexity of the algorithm was measured according to the guidelines in [28], the results are presented in Table V, the values of time are in seconds. *T0* is the time needed for computing the program given in [28] carrying out 1×10^6 times of the prescribed sequence of arithmetic operations, *T1* is the time of 2×10^5 evaluations of Function 18 and $\hat{T}2$ is the average time of five runs of the algorithm with MaxFES $= 2 \times 10^5$ on Function 18.

C. Parameters

This section is required in [28] in the prescribed structure.

- The values of the parameters controlling the competition remain on their default setting, that is $n_0 = 2$ and $\delta = 1/(H * 5) = 0.0167$.
- No dynamic range of parameter value is applied.
- Population size *NP* was set up to $NP = 50$ in all the experiments.
- The parameters controlling the condition of the restart, see (7), were set up to $\varepsilon_f = 1 \times 10^{-8}$ and $\varepsilon_d = 1$. The restart with a new initialization of the population is done if

$$f_{max} - f_{min} < 1 \times 10^{-8} \quad \text{AND} \quad \mathit{maxdist} < 1.$$

- Computation time needed to the tuning of the parameters can be estimated by the $6 \times$ *MaxFES* function evaluations per problem. A quick analysis of the results of tuning took about two hours.

D. Impact of Restart

The impact of restart on the efficiency of the algorithm was analyzed. The means of counts of restart for the test problems, where the mean value is positive at least in one dimension of the problem, are presented in Table VIII. In order to make reading easier, "-" is used instead of zero. It is obvious that there are many test problems, where no restart occurs, namely in the problems of higher dimension.

The impact of the restart on the efficiency of the algorithm was assessed by Wilcoxon two-sample tests, where the *b6e6rl* algorithms with and without restart are compared for the test problems. Only the problems with average count of restarts greater than one half are included into the comparison by Wilcoxon test. The results of the comparison are depicted in Table IX. The cases when the algorithm with restart is significantly better are denoted by "+" if the null hypothesis is rejected at $\alpha = 0.05$, "++" at $\alpha = 0.01$, and "+++" at $\alpha = 0.001$. The cases when the algorithm with restart is significantly worse are denoted by "- - -" signs. The symbol "=" means no significant difference. For $D = 10$, the restart is beneficial in eight benchmark problems, while only in one problem for $D = 30$. For $D = 50$ and $D = 100$, the restart is beneficial in three problems but the restart decreases the

performance in one and two problems, respectively. It indicates that the restart could be helpful but the counts of restart are not sufficient in some problems while the use of restart causes the deteriorating of the search in a few problems. A study how to recognize the proper conditions for the restart should continue.

TABLE VIII. MEANS OF COUNTS OF RESTARTS

F	$D = 10$	$D = 30$	$D = 50$	$D = 100$
1	3.10	-	-	-
2	3	3	-	-
3	4.35	4	-	-
4	4.98	0.06	0.45	0.06
6	1.92	-	-	-
7	0.41	4.57	4	3.18
8	4	4	4	3.65
10	2.94	2	2	2
13	0.10	0.10	0.08	-
14	0.10	0.02	0.08	-
17	1.35	-	-	-
18	0.29	-	0.04	-
20	0.12	-	-	-
21	0.92	-	-	-
23	6.06	5.06	6	3.02
24	-	1.35	3.37	3
25	1.75	2.35	0.92	1.71
26	0.08	0.20	-	-
27	0.80	0.31	-	-
28	3.22	0.33	0.06	-
29	1.84	-	-	-
30	1.67	0.37	0.06	-

TABLE IX. COMPARISON OF THE ALGORITHMS WITH AND WITHOUT CONTROLLED RESTART

F	$D = 10$	$D = 30$	$D = 50$	$D = 100$
1	=			
2	=	=		
3	=	=		
4	+++			
6	=			
7		=	- - -	- - -
8	=	=	=	- - -
10	+++	=	+++	++
17	++			
21	=			
23	=	=	=	=
24		=	+++	+++
25	+++	+++	+++	+++
27	++			
28	+++			
29	+++			
30	++			

VII. CONCLUSIONS

A new variant of competitive differential evolution with the controlled restart was proposed and tested on the CEC 2014 benchmark problems. The results show that the algorithm performs relatively well in the problems of lower dimension or in the problems with unrotated objective functions. However, systematical failure was observed in the minimization of several functions at all four levels of dimension, mostly in the problems with rotated objective functions.

We can conclude that the possible cause of the algorithm failure on rotated functions is the lack of such a strategy in the pool of competing strategies that is able to cope well with the minimization of rotated functions. Finding a new DE-strategy pool which makes the algorithm more efficient in the optimization problems with rotated objective functions is the topic for further research.

ACKNOWLEDGMENT

This work was supported by the European Regional Development Fund in the IT4Innovations Centre of Excellence project (CZ.1.05/1.1.00/02.0070) and partially supported by University of Ostrava from the project SGS15/PřF/2014.

REFERENCES

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

[2] ——, "Differential evolution – a simple and efficient adaptive scheme for global optimization over continuous spaces," Tech. Rep., 1995, TR-95-012. [Online]. Available: http://www.icsi.berkeley.edu/ storn/litera.html

[3] R. Storn, K. Price, and J. Lampinen, *Differential evolution – A practical approach to global optimization*. Berlin, Germany: Springer, 2005.

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

[5] 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.

[6] P. Kaelo and M. M. Ali, "A numerical study of some modified differential evolution algorithms," *European J. Operational Research*, vol. 169, pp. 1176–1184, 2006.

[7] D. Zaharie, "Influence of crossover on the behavior of differential evolution algorithms," *Applied Soft Computing*, vol. 9, pp. 1126–1138, 2009.

[8] 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.

[9] 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, pp. 398–417, 2009.

[10] 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.

[11] 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.

[12] Y. Wang, Z. Cai, and 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] R. Tanabe and A. Fukunaga, "Evaluating the performance of shade on cec 2013 benchmark problems," in *IEEE Congress on Evolutionary Computation 2013 Proceedings*, 2013, pp. 1952–1959.

[14] J. Brest, B. Boškovič, A. Zamuda, I. Fister, and E. Mezura-Montes, "Real parameter single objective optimization using self-adaptive differential evolution algorithm with more strategies," in *IEEE Congress on Evolutionary Computation 2013 Proceedings*, 2013, pp. 377–383.

[15] A. Zamuda, J. Brest, and E. Mezura-Montes, "Structured population size reduction differential evolution with multiple mutation strategies on cec 2013 real parameter optimization," in *IEEE Congress on Evolutionary Computation 2013 Proceedings*, 2013, pp. 1925–1931.

[16] I. Poikolainen and F. Neri, "Differential evolution with concurrent fitness based local search," in *IEEE Congress on Evolutionary Computation 2013 Proceedings*, 2013, pp. 384–391.

[17] S. Biswas, S. Kundu, S. Das, and A. V. Vasilakos, "Teaching and learning based differential evoltuion with self adaptation for real parameter optimization," in *IEEE Congress on Evolutionary Computation 2013 Proceedings*, 2013, pp. 1115–1122.

[18] L. D. S. Coelho, H. V. H. Ayala, and R. Z. Freire, "Populations variance-based adaptive differential evolution for real parameter optimization," in *IEEE Congress on Evolutionary Computation 2013 Proceedings*, 2013, pp. 1672–1677.

[19] F. Caraffini, F. Neri, J. Cheng, G. Zhang, L. Picinali, G. Iacca, and E. Mininno, "Super-fit multicriteria adaptive differential evolution," in *IEEE Congress on Evolutionary Computation 2013 Proceedings*, 2013, pp. 1678–1685.

[20] S. M. Elsayed, R. A. Sarker, and T. Ray, "Differential evolution with automatic parameter configuration for solving the CEC 2013 competition on real-parameter optimization," in *IEEE Congress on Evolutionary Computation 2013 Proceedings*, 2013, pp. 1932–1937.

[21] J. Lampinen and I. Zelinka, "On stagnation of differential evolution algorithm," in *MENDEL 2000, 6th International Conference on Soft Computing*. Brno: University of Technology, 2000, pp. 76–83.

[22] J. Tvrdík and R. Poláková, "Competitive differential evolution applied to cec 2013 problems," in *IEEE Congress on Evolutionary Computation 2013 Proceedings*, 2013, pp. 1651–1657.

[23] S. M. Elsayed and R. A. Sarker, "Differential evolution with automatic population injection scheme for constrained problems," in *IEEE Symposium on Differential Evolution (SDE) 2013 Proceedings*, 2013, pp. 112–118.

[24] J. Tvrdík, "Competitive differential evolution," in *MENDEL 2006, 12th International Conference on Soft Computing*, R. Matoušek and P. Ošmera, Eds. Brno: University of Technology, 2006, pp. 7–12.

[25] ——, "Adaptation in differential evolution: A numerical comparison," *Applied Soft Computing*, vol. 9, pp. 1149–1155, 2009.

[26] ——, "Self-adaptive variants of differential evolution with exponential crossover," *Analele of West University Timisoara, Series Mathematics-Informatics*, vol. 47, pp. 151–168, 2009.

[27] J. Tvrdík, R. Poláková, J. Veselský, and P. Bujok, "Adaptive variants of differential evolution: Towards control-parameter-free optimizers," in *Handbook of Optimization*. Springer, 2012, pp. 423–449.

[28] J. J. Liang, B. Qu, and P. N. Suganthan, "Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization," Nanyang Technological University, Singapore, 2013. [Online]. Available: http://www.ntu.edu.sg/home/epnsugan/