Proceedings of the International Multiconference on
Computer Science and Information Technology pp. 909–915

ISBN 978-83-60810-27-9
ISSN 1896-7094

Enhanced Competitive Differential Evolution for Constrained Optimization

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

Radka Poláková
University of Ostrava
Department of Mathematics
Ostrava, Czech Republic
Email: radka.polakova@wo.cz

***Abstract*—The constrained optimization with differential evolution (DE) is addressed. A novel variant of competitive differential evolution with a hybridized search of feasibility region is proposed, where opposition-based optimization and adaptive controlled random search are combined. Various variants of the algorithm are experimentally compared on the benchmark set developed for the special session of IEEE Congress of Evolutionary Computation (CEC) 2010. The results of the enhanced competitive DE show effective search of feasible solutions, in difficult problems significantly better than the competitive DE variant presented at CEC 2010.**

I. Introduction

DIFFERENTIAL evolution (DE) was proposed by Storn and Price [1] as a global optimizer for continuous optimization problems with a real-value objective function, where only the search space (domain) S is specified by lower (a_j) and upper (b_j) limits of each component j, $S = \prod_{j=1}^{D}[a_j, b_j]$, $a_j < b_j$, $j = 1, 2, \ldots, D$, D is the dimension of the domain. The global minimum point $\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}\in S} f(\boldsymbol{x})$ is the solution of the problem.

DE has been studied intensively in recent years and many new variants of DE have been proposed. The research has mostly focused on proper settings of control parameters and their adaptation during the search process. However, variants of DE for solving discrete or constrained problems have also appeared, for overview see e.g. [2].

The paper is organized as follows. In Section II, the constrained optimization problem is defined and sevral stochastic algorithms for its solution are mentioned. The DE algorithm is described in Section III, the description is focused on important features of the competitive DE used in this study. Techniques of enhanced search of the feasibility region are summarized and all algorithms tested in this study are described in Section IV. Section V involves specification of experiments and the results of comparison of the feasibility rates obtained by all the competitive DE variants in the tests as well as detailed results of the best performing algorithm. Concluding remarks are made in Section VI.

This work was supported by Czech Ministry of Education, the research project MSM 6198898701.

II. Constrained Optimization

We consider the optimization problem in the following format [3]:

$$\textit{Minimize: } f(\boldsymbol{x}),\ \boldsymbol{x} = (x_1, x_2, \ldots, x_D) \textit{ and } \boldsymbol{x} \in S \qquad (1)$$

$$\textit{subject to: } \begin{array}{ll} g_i(\boldsymbol{x}) \le 0, & i = 1, \ldots, p \\ h_j(\boldsymbol{x}) = 0, & j = p+1, \ldots, m. \end{array}$$

A solution is regarded feasible if $g_i(\boldsymbol{x}) \le 0$, *for* $i = 1, \ldots, p$, and $|h_j(\boldsymbol{x})| - \varepsilon \le 0$, *for* $j = p+1, \ldots, m$. Mean violation $\bar{v}$ of any point $\boldsymbol{x}$ in population defined according to [3] can be evaluated

$$\bar{v} = \frac{\sum_{i=1}^{p} G_i(\boldsymbol{x}) + \sum_{j=p+1}^{m} H_j(\boldsymbol{x})}{m},$$

where

$$G_i(\boldsymbol{x}) = \begin{cases} g_i(\boldsymbol{x}) & \text{if} \quad g_i(\boldsymbol{x}) > 0 \\ 0 & \text{if} \quad g_i(\boldsymbol{x}) \le 0 \end{cases}$$

$$H_j(\boldsymbol{x}) = \begin{cases} |h_j(\boldsymbol{x})| & \text{if} \quad |h_j(\boldsymbol{x})| - \varepsilon > 0 \\ 0 & \text{if} \quad |h_j(\boldsymbol{x})| - \varepsilon \le 0. \end{cases}$$

Some versions of DE for constrained problems have been published recently. One of the first modifications of DE for constrained problems was published by Lampinen [4]. In [5], the authors proposed self-adaptive algorithm ε-jDE which is a modification of jDE algorithm [6] extended for constrained optimization. They presented ε level controlling. The ε level is updated according to the mean violation of individuals in the current generation until the number of generations G reaches the control generation G_C. After the number of generations exceeds G_C, the ε level is set to 0 to obtain a solution with minimum constraint violation.

An up-to-date overview of evolutionary techniques for constrained problems is presented in [7]. The authors also proposed a novel algorithm using an ensemble of constraint handling techniques (ECHT). The included techniques are superiority of feasible solution, self-adaptive penalty, ε-constraint, and stochastic ranking. In the algorithm which uses ECHT together with differential evolution (ECHT-DE) each constraint handling method has its own population and new trial points are computed separately for each constraint handling method in the current generation.

A simple DE algorithm for constrained problem was presented in [8]. This algorithm alternates minimization of mean violation $\bar{v}$ and $f(\boldsymbol{x})$ using one generation of the adaptive DE algorithm with 12 competing DE strategies, where various values of control parameters and two types of crossover are used. The algorithm performed well on most problems of the benchmark test set [3]. However, in a few problems the algorithm found feasible solution rarely or even never, see counts of feasible solutions in Tables II and III. This failure in search of the feasible region is a motivation for the proposal of a new algorithm with enhanced search of the feasible region.

III. Differential Evolution

Algorithm of DE works with a population of individuals (*NP* points in domain S) that are considered as candidates of solution. The population develops iteratively by using evolutionary operators of selection, mutation, and crossover. Each iteration corresponds to an evolutionary generation. Let us denote two subsequent generations P and Q. Application of evolutionary operators in the old generations P creates a new generation Q. After completing the new generation Q, the Q becomes the old generation for next iteration. The basic scheme of DE is written in a pseudo-code in Figure 1.

```
1   generate an initial generation P, (x_i, i = 1,2,...,NP)
2   while stopping condition not achieved
3      for i := 1 to NP do
4         generate a new trial vector y using P
5         if  y  better x_i then   insert y into Q
6                          else   insert x_i into Q
7         endif
8      endfor
9      P := Q
10  endwhile
```

Fig. 1. **Algorithm 1** – Basic scheme of DE in pseudo-code

The relation "better" (line 5 in Algorithm 1) means $f(\boldsymbol{y}) < f(\boldsymbol{x}_i)$ in unconstrained problems. In constrained problems, the preference of a more feasible solution is usually applied according to the following rule: Accept a new trial point $\boldsymbol{y}$ if

$$\bar{v}_y < \bar{v}_i \text{ or } (\bar{v}_y = 0 \text{ and } \bar{v}_i = 0 \text{ and } f(\boldsymbol{y}) < f(\boldsymbol{x}_i)), \quad (2)$$

where $\bar{v}_i$ is mean violation of the current point $\boldsymbol{x}_i$.

A new trial point $\boldsymbol{y}$ (line 4 in Algorithm 1) is generated by using mutation and crossover. There are various strategies of mutation and crossover [1], [2], [9]. The most popular mutation strategy called DE/rand/1/ generates the mutant point $\boldsymbol{u}$ by adding the weighted difference of two points,

$$\boldsymbol{u} = \boldsymbol{r}_1 + F\,(\boldsymbol{r}_2 - \boldsymbol{r}_3)\,, \quad F > 0\,, \quad (3)$$

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. In the implementation of the DE algorithm in this paper is used a slightly modified mutation strategy. It was proposed by Kaelo and Ali [10]. They called it random localization and it can be denoted as DE/randrl/1/. The point $\boldsymbol{r}_1$ in (3) is the best one among $\boldsymbol{r}_1$, $\boldsymbol{r}_2$, and $\boldsymbol{r}_3$.

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

$$y_j = \begin{cases} u_j & \text{if} \quad U_j \le CR \quad \text{or} \quad j = l \\ x_{ij} & \text{if} \quad U_j > CR \quad \text{and} \quad j \ne l\,, \end{cases} \quad (4)$$

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 $CR \in [0, 1]$ is an input parameter influencing the number of elements to be exchanged by the crossover. The rule given by Eq. (4) ensures that at least one element of vector $\boldsymbol{x}_i$ is changed, even if $CR = 0$. The DE variants applying binomial crossover according to (4) are denoted DE/$\cdot$/$\cdot$/bin in literature.

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

The probability of crossover p_m can be defined as the mean relative length of overwritten elements of $\boldsymbol{x}_i$, i.e. $p_m = E(L)/D$. The relation between the p_m and control parameter *CR* was studied by Zaharie [11], [12]. For the binomial crossover, the relation between p_m and control parameter *CR* is linear,

$$p_m = CR(1 - 1/D) + 1/D, \quad (5)$$

while for the exponential crossover the relationship is strongly non-linear,

$$p_m = \frac{1 - CR^D}{D(1 - CR)}, \quad \text{for} \quad CR < 1. \quad (6)$$

This non-linearity should be taken into account when we set the *CR* value for a problem to be solved.

Differential evolution has a few control parameters, 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 [1], [2], [9], [13], [14] but their applicability is not universal.

Because trial-and-error control-parameter tuning is time-consuming, several new adaptive variants of DE have been recently proposed, e.g. [6], [15]–[21].

Competitive setting of the control parameters was proposed in [22]. In this adaptive approach to the choice of proper DE strategy, we randomly choose among H different settings of control parameters (F, *CR*, mutation, and crossover) with probabilities $q_h,\ h = 1, 2, \ldots, H$.

These probabilities change according to the success rate of the settings in the preceding steps of the search process. The h-th 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, see line 5 of Algorithm 1 in Figure 1. Probability q_h is evaluated as the relative frequency

$$q_h = \frac{n_h + n_0}{\sum_{j=1}^{H}(n_j + n_0)}\ , \tag{7}$$

where n_h is the current count of the h-th 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 h-th parameter setting. To avoid degeneration of the search process, the current values of q_h are reset to their starting values $q_h = 1/H$ if any probability q_h decreases bellow the given limit $\delta > 0$ during the search process.

Mutation according to (3) could cause that a new trial point $\boldsymbol{y}$ moves out of the domain S. In such a case, the point $\boldsymbol{y}$ can be skipped and a new one generated. A more effective attempt is to replace the values of $y_j < a_j$ or $y_j > b_j$, either by random value in $[a_j, b_j]$, see [9], or by reversed value of this coordinate y_j into S over the a_j or b_j by mirroring, see [23]. The latter method is used in algorithms implemented for numerical tests.

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

	$D = 10$				$D = 30$		
i	1	2	3		1	2	3
p_{mi}	0.3250	0.5500	0.7750	p_i	0.2750	0.5167	0.7583
CR_i	0.7011	0.8571	0.9418	CR_i	0.8815	0.9488	0.9801

In several benchmark tests [24], [25], the competitive variant of DE/randrl/1/ with $H = 12$ competing settings was among the most efficient. In this variant, six various settings of (F, CR) for the binomial crossover and six settings for the exponential crossover take part in the competition. Two values of F are used for the both type of crossover, one of them $F = 0.5$ is rather small and the second one $F = 0.8$ is rather big when the recommendation in [1], [2], [9], [13], [14] are followed. Three values of *CR* for the binomial crossover are set to their minimum, middle, and maximum value, that is $CR = 0$, $CR = 0.5$, and $CR = 1$. All the three values are combined in tuples with two values of F, which gives six settings mentioned above. For the exponential crossover, three values of *CR* are used as well, but they are set in such a way in order to have the values of the crossover probability given by (6) equally spaced inside the interval of their possible values, $[1/D,\ 1]$. Notice, that for extremal values of the crossover probability $p_m = 1/D$ or $p_m = 1$ the both types of crossover do not differ. Both the mutation probability values and the corresponding *CR* values are dependent on the dimension of the problem. The values of mutation probability and the corresponding values of *CR* evaluated from (6) for $D = 10$ and $D = 30$ are given in Table I. Like in the case of the binomial crossover, three values of *CR* are combined with two values of F, which gives six settings of (F, CR) for the exponential crossover taking part in competition. The competitive DE variant using these 12 settings is hereafter referred as *b6e6rl*.

Thus, the *b6e6rl* algorithm modified for constrained problems has only the following control parameters:

- Size of the population, *NP* – can be intuitively set with respect to the dimension of the problem.
- Parameters n_0 and δ to control the competition – can be set to the values, $n_0 = 2$, $\delta = 1/(5 \times H)$ used in other problems.
- Stopping condition – can be formed by maximum of objective function evaluations (MaxFES) or in other way.
- Parameter ε for the tolerance of feasibility violation – the value depends on the user's request.

IV. ENHANCED SEARCH OF FEASIBILITY REGION

The basic idea, how to enhance the search of the feasibility region, consists in exploitation of strategies different from DE in the search. Opposite-based optimization and adaptive controlled random search were applied for minimization of mean violation $\bar{v}$ in this study.

A. Opposition-Based Optimization

Application of opposition-based learning in stochastic optimization algorithms has appeared recently [26], [27] and it is called opposition-based optimization (OBO). Basic idea is to search for solution not only within the individuals of the population developed by evolutionary operators but also in the opposite part of the search space T. The opposite point to the point $\boldsymbol{x} \in T$, $T = \prod_{j=1}^{D}[l_j, u_j],\ l_j < u_j,\ j = 1, 2, \ldots, D$ is defined as symmetric with respect to the center of T by

$$\breve{\boldsymbol{x}} = \boldsymbol{l} + \boldsymbol{u} - \boldsymbol{x}. \tag{8}$$

The opposite population O of *NP* points to the population P according to relation (8) 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, $T = S$, which is $\boldsymbol{l} = \boldsymbol{a}$ and $\boldsymbol{u} = \boldsymbol{b}$. The search space T for the opposite points shrinks dynamically during the search process. If P is a matrix of size (*NP*, D), then $\boldsymbol{l} = \min(P)$ and $\boldsymbol{u} = \max(P)$, the functions $\min(\cdot)$ and $\max(\cdot)$ return the vectors of column minima and maxima, respectively. Thus, current dynamically shrinking search space for the opposite population can be

written as

$$T = \prod_{j=1}^{D}[l_j, u_j],\ l_j < u_j,\ j = 1, 2, \ldots, D, \quad \text{where} \quad \boldsymbol{l} = \min(P), \quad \boldsymbol{u} = \max(P). \tag{9}$$

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 $rand(0,1) < JR$, where $rand(0,1)$ is a random value from uniform distribution on $[0,1)$ and *JR* (jumping rate) is an input parameter of the algorithm. The jumping rate is usually set to constant value, e.g. $JR = 0.3$ [26].

B. Adaptive Controlled Random Search

Another algorithm for enhanced search of the feasible solution is the adaptive controlled random search (CRS) with four competing local heuristics. This algorithm was applied successfully to the estimation of parameters in non-linear regression models [28].

Three of the competing local heuristics are based on a randomized reflection in the simplex Σ created by $D+1$ points chosen from P. A new trial point $\boldsymbol{y}$ is generated from the simplex by the relation

$$\boldsymbol{y} = \boldsymbol{g} + U\left(\boldsymbol{g} - \boldsymbol{x}_H\right), \tag{10}$$

where $\boldsymbol{x}_H = \arg\max_{\boldsymbol{x}\in\Sigma} f(\boldsymbol{x})$ and $\boldsymbol{g}$ is the centroid of remaining D points of the simplex Σ. The multiplication factor U is a random variable distributed uniformly in $[s, \alpha - s)$, $\alpha > 0$ and s being input parameters, $0 < s < \alpha/2$. All the $D+1$ points of simplex are chosen randomly from P in two heuristics. Regarding the third heuristic, one point of the simplex is the point of P with the minimum objective function value and the remaining D points of the simplex Σ are chosen randomly from the remaining points of P. The fourth competing local heuristic is derived from differential evolution, where strategy DE/rand/1/bin is used and adaptive setting of control parameter F according to [15] is applied. The algorithm was used as described in [28] with the same setting of the parameters controlling the competition of local heuristics.

C. Enhanced Search of Feasible Individuals

The scheme of enhanced search of the feasible region is written in a pseudo-code in Figure 2. By the term "one generation" of an algorithm is meant one generation of DE (algorithm *b6e6rl*) or a run of another algorithm, where *NP* evaluations of objective function is needed. Notice that except the *b6e6rl* all the other algorithms minimize only the mean violation $\bar{v}$ as objective function. In the algorithm described in Figure 2 (hereafter denoted *CRSrOBO*) are shown all the enhanced search techniques used in this study. *CRSrOBO* appeared best performing in the experimental tests. The other algorithms tested miss one or more enhancement parts. The algorithm labeled by *CRSOBO* does not use the refreshment of population for the controlled random search, i. e. lines 4 to 6 in Figure 2 are not applied. The algorithm not using lines 4 to 6 and no opposition-based optimization (lines 10 to 12 Figure 2) is labeled *CRS*. The algorithm without enhancing the search of the feasible solutions by controlled random search (skipped lines 3 to 9 in Figure 2) is marked as *OBO*. The dynamically shrinking search space according to relation (9) is applied in all the algorithms using enhanced search by OBO, whereas the static search space S is used in refreshment of the population for CRS, because of no a priori information on allocation of the feasibility regions is assumed. One generation of OBO with static search space S is also applied immediately after the initialization of P at line 1 in Figure 2.

```
1  generate an initial generation P
2  while stopping condition not achieved
3     if no individual feasible then
4        if rand(0,1) < JR_CRS then
5           refresh P using opposite population in S
6        endif
7        perform one generation of CRS
8     endif
9     perform one generation of b6e6rl
10    if no individual feasible & rand(0,1) < JR_OBO then
11       perform one generation of OBO
12    endif
13    P := Q
14 endwhile
```

Fig. 2. **Algorithm 2** – Enhanced Search of Feasible Solutions

The mnemonic labels of the algorithms with enhanced search respect the applied techniques and the basic algorithm of competitive DE is not stressed in labels of variants because it is present in all the tested algorithms. Finally, if we omit all enhanced search (lines 3 to 8 and 10 to 12) then the rest is the algorithm *b6e6rl* described in Section III.

V. Experiments and Results

The modification of DE for the solution of constrained problems presented in [8] consists in alternating minimization of mean violation $\bar{v}$ and $f(\boldsymbol{x})$ using one generation of the *b6e6rl* algorithm. An extra minimization of mean violation could be considered as enhanced search of the feasibility region by *b6e6rl*. This variant labeled by *CEC10* hereafter is compared with the algorithms described in this study. All the variants of algorithm were implemented in Matlab.

The algorithms were tested on the benchmark set developed for the special CEC 2010 session [3]. The set consists of 18 problems at two levels of dimension. The code for test-function evaluation was loaded from the web page of the organizers [3].

Six algorithms were compared in experimental tests, namely algorithm labeled *CEC10*, simple competitive DE for constrained optimization (*b6e6rl*) and four new variants with enhanced search of the feasible solutions (*CRS*, *OBO*, *CRSOBO*, and *CRSrOBO*).

The control parameters of the algorithms were set to the same values for all the test problems, some parameters depend on the dimensionality of the test problems, the values follows:

- Size of the population, $NP = 50$ for $D = 10$, $NP = 100$ for $D = 30$.
- Parameters to control the competition, $n_0 = 2$, $\delta = 1/(5 \times H)$ for all the test problems.
- Stopping condition, MaxFES $= 2 \times 10^5$ for $D = 10$, MaxFES $= 6 \times 10^5$ for $D = 30$, as given in [3].
- Parameter for tolerance of feasibility violation, the value $\varepsilon = 0.0001$ is prescribed in [3].
- Jumping rate for OBO was set to $JR_{OBO} = 0.3$ as mostly applied in [27].
- Jumping rate of population refreshment for CRS was set to $JR_{CRS} = 0.1$ in order not to waste time by too frequent using static OBO.
- Control parameters of adaptive CRS were used the same as in [28].

No tuning of control-parameter was performed before the test experiments.

For each problem and dimension, 25 independent optimization runs were performed. From each run the function value $f(\boldsymbol{x}^*)$ found at the minimum of the mean violation $\bar{v}$ is returned together with the corresponding values of variables needed for the processing of the results presented in the tables. If there are more feasible solutions in the last generation, the minimum function value $f(\boldsymbol{x}^*)$ of the feasible solutions is returned.

TABLE II
COUNT OF FEASIBLE SOLUTIONS OF 25 FOR $D = 10$.

Problem	CEC10	b6e6rl	CRS	OBO	CRSOBO	CRSrOBO
1	25	25	25	25	25	25
2	25	25	24	18	22	21
3	25	25	25	24	25	25
4	25	25	25	25	25	25
5	4	0	0	16	14	16
6	2	0	1	23	21	20
7	25	25	25	25	25	25
8	25	25	25	25	25	25
9	3	2	0	12	18	22
10	3	5	3	11	20	17
11	20	0	22	0	24	24
12	25	16	25	21	25	25
13	25	25	25	25	25	25
14	25	25	25	25	25	25
15	24	25	25	22	24	24
16	19	25	20	25	23	24
17	19	21	21	19	23	21
18	22	24	25	16	23	20
Average	18.94	17.67	18.94	19.83	22.89	22.72

Comparison of the feasibility rates obtained by all the tested variants are shown in Tables II and III, where counts of runs that found a feasible solution ($\bar{v} = 0$) of 25 runs are given. If we compare the results of the algorithms in problems 5, 6, and 9 to 12, we see a complementary effect of different enhanced techniques. Where CRS does not help, OBO is useful, and vice versa. Moreover, a synergic effect appears if more types of enhanced search are applied together. The best results were obtained by *CRSrOBO*, where the average count of the feasible solutions is almost 23 of 25 and the minimum of feasible solutions is 14, i.e. more than a half of 25.

TABLE III
COUNT OF FEASIBLE SOLUTIONS OF 25 FOR $D = 30$.

Problem	CEC10	b6e6rl	CRS	OBO	CRSOBO	CRSrOBO
1	25	25	25	25	25	25
2	25	25	25	25	25	24
3	25	25	25	25	24	24
4	25	25	13	25	13	17
5	0	0	0	9	4	14
6	4	0	0	19	7	17
7	25	25	25	25	25	25
8	25	25	25	25	25	25
9	1	6	2	19	16	20
10	2	5	0	19	20	21
11	25	0	17	0	21	23
12	13	1	23	0	20	23
13	25	25	25	25	25	25
14	25	25	25	25	25	25
15	25	25	25	25	25	25
16	25	25	25	25	25	25
17	23	25	24	25	25	25
18	25	25	25	24	25	24
Average	19.06	17.33	18.28	20.28	20.83	22.61

The two-tailed Fisher's exact test was applied to the statistical comparison of the feasibility rates of the algorithms. All the algorithms were compared with respect to *CRSrOBO*. The values of the feasibility rate for which the proportion of the feasible solutions is different significantly from the corresponding proportion of *CRSrOBO* at level of 0.05 are underlined.

Detailed results of the best performing variant of enhanced competitive DE (*CRSrOBO*) in 18 test problems at two levels of the dimension are presented in Tables IV and V. The structure of the results follows the requirements given in [3]. The results obtained from 25 runs of each problem were sorted according to their values of mean violation $\bar{v}$ in an ascending way. Then the part of the feasible solutions was sorted according to their function values $f(\boldsymbol{x}^*)$ in an ascending way. The presented values of $f(\boldsymbol{x}^*)$ in columns Best, Median, and Worst refer to this order. The sorting according to the values of mean violation $\bar{v}$ can cause that the best or median of $f(\boldsymbol{x}^*)$ need not be necessarily less than the worst one in some of the problems. The values of $\bar{v}$ and c correspond to the row of Median. Three numbers in vector c indicate the number of violations more than 1, the number of violations in $(0.1,\ 1]$, and the number of violations in $(0.0001,\ 0.1]$. The mean and the standard deviation (Stdev) of $f(\boldsymbol{x}^*)$ are evaluated from all

TABLE IV
DETAILED RESULTS OF CRSRORBO ALGORITHM AFTER ACHIEVING MAXFES FOR $D = 10$.

Problem	Best	Median	Worst	c	$\bar{v}$	Mean	Stdev	FR
1	-0.74731	-0.74731	-0.74056	0,0,0	0	-0.74623	0.00253	100 %
2	-0.85975	3.0025	3.1052	0,0,0	0	2.5482	1.3519	84 %
3	0	0	5.180e+11	0,0,0	0	2.072e+10	1.036e+11	100 %
4	-0.00001	-0.00001	-0.00001	0,0,0	0	-0.00001	0	100 %
5	6.8037	241.51	533.08	0,0,0	0	205.13	167.99	64 %
6	-36.480	203.55	469.88	0,0,0	0	197.07	145.76	80 %
7	0	0	3.9866	0,0,0	0	0.31893	1.1038	100 %
8	0	3.6229	40.876	0,0,0	0	6.6088	8.9421	100 %
9	2.099e+10	2.905e+12	5.438e+12	0,0,0	0	3.895e+12	3.727e+12	88 %
10	6.005e+10	1.437e+12	3.676e+12	0,0,0	0	3.989e+12	4.606e+12	68 %
11	-0.00152	-0.00152	-0.08734	0,0,0	0	-0.00496	0.01716	96 %
12	-305.49	-0.19925	-0.19925	0,0,0	0	-26.310	69.644	100 %
13	-68.429	-68.429	-68.429	0,0,0	0	-68.429	7.234e-08	100 %
14	6.8326	3.197e+08	1.496e+10	0,0,0	0	1.680e+09	3.389e+09	100 %
15	3.876e+11	2.925e+13	1.012e+14	0,0,0	0	4.242e+13	5.102e+13	96 %
16	0	0.90857	0.92313	0,0,0	0	0.74910	0.32623	96 %
17	17.319	242.92	751.13	0,0,0	0	247.91	186.64	84 %
18	113.90	4379.01	10223.3	0,0,0	0	5823.3	4195.9	80 %

TABLE V
DETAILED RESULTS OF CRSRORBO ALGORITHM AFTER ACHIEVING MAXFES FOR $D = 30$.

Problem	Best	Median	Worst	c	$\bar{v}$	Mean	Stdev	FR
1	-0.82188	-0.82188	-0.81795	0,0,0	0	-0.82095	0.00169	100 %
2	2.0180	3.4494	4.1897	0,0,0	0	3.4143	0.63100	96 %
3	0	1.129e+08	9.749e+11	0,0,0	0	3.350e+12	5.015e+12	96 %
4	-3.263e-06	0.00191	0.92952	0,0,0	0	0.25626	0.42471	68 %
5	73.320	506.36	556.27	0,0,0	0	352.56	176.17	56 %
6	196.45	524.00	582.50	0,0,0	0	439.48	125.78	68 %
7	0	0	0	0,0,0	0	0	0	100 %
8	0	0	2.335e-26	0,0,0	0	1.081e-27	4.693e-27	100 %
9	2.937e+11	1.934e+13	2.445e+13	0,0,0	0	1.773e+13	1.672e+13	80 %
10	1.861e+11	1.822e+13	3.883e+13	0,0,0	0	1.869e+13	1.572e+13	84 %
11	-0.00039	-0.00039	0.01867	0,0,0	0	0.00113	0.00528	92 %
12	-0.19926	-0.19926	196.42	0,0,0	0	7.4997	39.368	92 %
13	-67.224	-65.433	-64.449	0,0,0	0	-65.617	0.74208	100 %
14	2.783e+08	3.130e+10	2.708e+11	0,0,0	0	4.778e+10	6.011e+10	100 %
15	1.111e+13	1.349e+14	2.599e+14	0,0,0	0	1.312e+14	7.212e+13	100 %
16	0.29150	0.80218	1.0364	0,0,0	0	0.78285	0.20885	100 %
17	86.084	781.38	1533.7	0,0,0	0	719.01	491.01	100 %
18	55.420	15580.9	42534.6	0,0,0	0	19026.4	14999.8	96 %

the 25 runs. The feasibility rate in the last column of tables is defined as

$$FR = 100 \times \frac{\text{number of runs with feasible solution}}{\text{number of all runs}}.$$

In comparison with the results of *CEC10* [8], not only the feasibility rate of *CRSrOBO* is significantly better in some problems, but also the minimum function values (column Best) found by *CRSrOBO* are less than those found by *CEC10* in more than a half of problems. Thus, the novel enhanced competitive DE variant combining adaptive controlled random search with population refreshment and opposition-based optimization for the search of feasible solution proved to be a substantial improvement of the competitive DE for constrained single-objective optimization.

VI. CONCLUSION

A novel competitive DE variant with enhanced search of the feasibility region for constrained single-objective optimization was proposed. This algorithm includes adaptive controlled random search with population refreshment by applying opposite

points in the search space and opposition-based optimization. These techniques are used for the search of the feasibility region (minimization of violation), whereas the competitive differential evolution is used both for minimizing the constraints' violation and the function value. The novel algorithm proved to be a substantial improvement of the competitive DE for constrained single-objective optimization presented recently [8].

In near future, we are planning to compare this algorithm with good performing algorithms available in literature [5], [7] and with the other algorithms succeeded in the competition of Congress on Evolutionary Computation 2010 at Barcelona. It is needed for a decision on applicability of the algorithm in real-world problems. A more detailed analysis of the algorithms is intended. Its results can bring a new impuls for further development of algorithms for constrained single-objective optimization with enhanced search of the feasible solutions.

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] K. V. Price, R. Storn, and J. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*. Springer, 2005.

[3] R. Mallipeddi and P. N. Suganthan, "Problem definition and evaluation criteria for the CEC 2010 competition and special session on single objective constrained real-parameter optimization," Tech. Rep., Nanyang Technological University, April, 2010. [Online]. Available: http://www3.ntu.edu.sg/home/EPNSugan/

[4] J. Lampinen, "A constrained handling approach for differential evolution algorithm," in *IEEE Congress on Evolutionary Computation*, 2002, pp. 1468–1473.

[5] J. Brest, "Constrained real-parameter optimization with ε-self-adaptive differential evolution," in *Constraint-Handling in Evolutionary Optimization*, E. Mezura-Montez, Ed. Springer, 2009, pp. 73–93.

[6] 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.

[7] R. Mallipeddi and P. N. Suganthan, "Ensemble of constraint handling techniques," *IEEE Transactions on Evolutionary Computation*, 2010, doi 10.1109/TEVC.2009.2033582.

[8] J. Tvrdík and R. Poláková, "Competitive differential evolution for constrained problems," in *IEEE Congress on Evolutionary Computation*, 2010, pp. 1632–1639.

[9] V. Feoktistov, *Differential Evolution in Search of Sotution*. Springer, 2006.

[10] 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.

[11] D. Zaharie, "A comparative analysis of crossover variants in differential evolution," in *Proceedings of IMCSIT 2007*, U. Markowska-Kaczmar and H. Kwasnicka, Eds. Wisla: PTI, 2007, pp. 171–181.

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

[13] R. Gämperle, S. D. Müller, and P. Koumoutsakos, "A parameter study for differential evolution," in *Advances in Intelligent Systems Fuzzy Systems, Evolutionary Computing*, A. Grmela and N. E. Mastorakis, Eds. Athens: WSEAS Press, 2002, pp. 293–298.

[14] D. Zaharie, "Critical values for the control parameter of the differential evolution algorithms," in *MENDEL 2002, 8th International Conference on Soft Computing*, R. Matoušek and P. Ošmera, Eds. Brno: University of Technology, 2002, pp. 62–67.

[15] M. M. Ali and A. Törn, "Population set based global optimization algorithms: Some modifications and numerical studies," *Computers and Operations Research*, vol. 31, pp. 1703–1725, 2004.

[16] J. Liu and J. Lampinen, "A fuzzy adaptive differential evolution algorithm," *Soft Computing*, vol. 9, pp. 448–462, 2005.

[17] A. K. Qin and P. N. Suganthan, "Self-adaptive differential evolution for numerical optimization," in *IEEE Congress on Evolutionary Computation*, 2005, pp. 1785–1791.

[18] M. G. H. Omran, A. Salman, and A. P. Engelbrecht, "Self-adaptive differential evolution," in *Lecture Notes in Artifitial Intelligence*, ser. 3801. Springer, 2005, pp. 192–199.

[19] A. Qin, V. Huang, and P. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Transactions on Evolutionary Computation*, vol. 13, pp. 398–417, 2009.

[20] J. Zhang and A. C. Sanderson, *Adaptive Differential Evolution, A Robust Approach to Multimodal Problem Optimization*. Springer, 2009.

[21] M. Weber, V. Tironen, and F. Neri, "Scale factor inheritance mechanism in distributed differential evolution," *Soft Computing*, vol. 14, pp. 1187–1207, 2010.

[22] 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.

[23] V. Kvasnička, J. Pospíchal, and P. Tiňo, *Evolutionary Algorithms*. Bratislava: Slovak Technical University, 2000, (In Slovak).

[24] J. Tvrdík, "Adaptive differential evolution and exponential crossover," in *Proceedings of IMCSIT 2008*, U. Markowska-Kaczmar and H. Kwasnicka, Eds. Wisla: PTI, 2008, pp. 927–931.

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

[26] S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, "Opposition-based differential evolution," *IEEE Transactions on Evolutionary Computation*, vol. 12, pp. 64–79, 2008.

[27] S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, "Opposition-based differential evolution," in *Advances in Differential Evolution*, U. K. Chakraborty, Ed. Springer, 2008, pp. 155–171.

[28] J. Tvrdík, I. Křivý, and L. Mišík, "Adaptive population-based search: application to estimation of nonlinear regression parameters," *Computational Statistics and Data Analysis*, vol. 52, pp. 713–724, 2007.