L-SHADE with Competing Strategies Applied to CEC2015 Learning-based Test Suite

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
(phone: +420 59709 1409;
email: radka.polakova@osu.cz).

Josef Tvrdík
Department of Computer Sciences
University of Ostrava,
30. dubna 22, 701 03 Ostrava,
Czech Republic
(phone: +420 59709 2183;
email: josef.tvrdik@osu.cz).

Petr Bujok
Department of Computer Sciences,
University of Ostrava,
30. dubna 22, 701 03 Ostrava,
Czech Republic
(phone: +420 59709 2147;
email: petr.bujok@osu.cz).

***Abstract*—Successful adaptive variant of differential evolution, the Success-history based parameter adaptation of Differential Evolution using linear population size reduction algorithm (L-SHADE), was improved. Adaptive mechanisms used in the algorithm were joined with adaptive mechanism proposed for competitive differential evolution algorithm. Four strategies, including the original one and strategies with exponential crossover, compete in the new LSHADE44 algorithm. The proposed algorithm is applied to the benchmark set defined for Learning-based case of Special Session and Competitions on Real-Parameter Single Objective Optimization on CEC2016. According to preliminary experiments, the proposed algorithm with competing strategies outperformed the original L-SHADE in the most of the test problems.**

I. Introduction

A new adaptive variant of differential evolution (DE) is introduced in this paper. The proposed algorithm is applied to CEC2015 Learning-based Test Suite [1] and the results are submitted to learning-based case of Special Session and Competitions on Real-Parameter Single Objective Optimization on CEC2016 in order to compare the performance of the newly proposed DE variant with the current state-of-the-art optimization algorithms. The results of proposed algorithm have been submitted also to another part of the competition called single parameter-operator set based case. Due to this fact and due to the competition organizers' request of separate paper for each part of competition when an algorithm participates in more than one part of competition, this paper is similar as the paper submitted there in some parts, mainly the introduction and the parts where the proposed algorithm is described.

The global optimization problem with boundary constraints is considered. The real function $f : R^d \to R$ and the search space $S = \prod_{j=1}^{d}[a_j, b_j] \subset R^d$, $a_j < b_j, j = 1, 2, \ldots, d$, are defined. We are searching for a point x^* such that $f(x^*) \leq f(x)$, $\forall x \in S$, d is the dimension of the problem, f is an objective function. Therefore, the global minimum point of function f in the search space S is to be found.

Differential evolution [2] is an evolutionary algorithm for solving the global optimization problems. It is a stochastic algorithm, which does not guarantee to find the precise solution in each run. The algorithm works with a population of points from S. Each member of the population is a potential solution of the problem. At the beginning of the search, the points of the population are chosen randomly from the uniform distribution in the search space S. The points (and therefore also population) evolve from a generation to a generation.

Differential evolution algorithm is described by pseudo-code in Algorithm 1. P is the population of points, N is the size of population.

Algorithm 1 Differential evolution

```
generate an initial population P = (x_1, x_2, ..., x_N),
  x_i ∈ S distributed uniformly
evaluate f(x_i), i = 1, 2, ..., N
while stopping condition not reached do
  Q = ∅
  for i = 1 to N do
    generate a trial vector y
    evaluate f(y)
    if f(y) ≤ f(x_i) then
      insert y into new generation Q
    else
      insert x_i into new generation Q
    end if
  end for
  P := Q
end while
```

In each generation, so called trial point is generated for each point of the current population and its value of the objective function is computed. If the value of objective function in the trial point is better or equal to the value in corresponding current point, this new trial point replaces the original point in the new generation of the population. The population evolves until the stopping condition is not reached. The best point of the final generation of population is the solution of problem found by DE.

Two evolutionary operations, namely a mutation and a

978-1-5090-0623-6/16/$31.00

crossover, are used in generating a new trial point (line 6 in Algorithm 1). Each of these two evolutionary operations can be done by several ways. The new trial point is developed by using some points of the current generation of population. Firstly, a mutant $\boldsymbol{u}$ is created by a chosen variant of mutation operation. Then the trial point $\boldsymbol{y}$ is constructed by mixing coordinates of the current point $\boldsymbol{x}_i$ with coordinates of mutant $\boldsymbol{u}$.

Differential evolution algorithm has several control parameters and the effectiveness of the algorithm when solving a particular optimization problem strongly depends on their values. Parameters of DE are: size of population, strategy of producing trial point, and stopping condition. Additional parameters of DE algorithm are numeric parameters F and *CR*, which control the mutation and the crossover. It is expected that adaptation of these parameters is beneficial for successful solving the different types of optimization problems.

The rest of the paper is organized as follows. Section II shows background and motivation for the proposal of a new combined adaptive mechanism. The proposed algorithm is described in the Section III. Experimental settings are specified in Section IV. The experimental results are presented in Section V. Finally, some concluding remarks are given in the last section.

II. Motivation for Proposed Algorithm

There are many adaptive versions of differential evolution proposed in literature. The CEC competitions promote the development of new adaptive algorithms and their comparison on common hard test problems.

One of the oldest proposed adaptive version of DE is jDE algorithm [3], where the only one strategy for providing of a trial point (rand/1/bin) is used and the parameters F and *CR* are self-adapted by their evolution with the population. The size of population remains the same during the whole process.

The parameters F and *CR* and the strategy of the current trial point creation are also adapted in other versions of DE, e.g. SADE [4], CoDE [5], EPSDE [6], and CDE [7] .

CDE algorithm is based on competition of H strategies. Each of H strategies in the pool is used for generating the new trial point with the probability proportional to its success in preceding period of the search. One of the most effective version of CDE is the *b6e6rl* version. The version employs two types of crossover proposed for differential evolution together with mutation randrl/1, which is an enhanced modification of the most widely used rand/1 mutation operator.

In JADE [8] algorithm, only current-to-pbest/1/bin strategy is used for creation of the trial points. The parameters F and *CR* are here generated from Cauchy and normal distribution, respectively. The parameters of the distributions are estimated from successful values of F and *CR* in last generation. The current-to-pbest/1 mutation is derived from current-to-best/1 mutation known before. However, it is less greedy than original mutation. The algorithm uses also an archive, in which old replaced members of the population are stored.

The SHADE algorithm [9], [10] inspired by JADE was successful in CEC2013 competition [11], [12]. The current-to-pbest/1/bin strategy is employed in SHADE algorithm and parameters F and *CR* are generated from Cauchy and normal distribution like in JADE algorithm. In contrast to JADE, F and *CR* are generated by using successful values of these parameters from several last generations with using the pair of so called historical circle memories in SHADE algorithm. The size of population is not adapted in SHADE algorithm.

Later, the authors of SHADE algorithm included the idea of linearly decrease of population size during the search into SHADE and they proposed L-SHADE algorithm [13] that was successful on CEC2014 competition [14], [15].

The adaptive mechanism in SHADE4 algorithm [16] inherits all the features of SHADE algorithm. Additionally, it uses the competition mechanism and strategies of *b6e6rl* and also current-to-pbest/1/exp strategy. The use of the strategies with the exponential crossover was inspired by the results of [17], [18], [19], [20].

III. L-SHADE Algorithm with Competing Strategies

The adaptive mechanisms used in the proposed algorithm are described in detail. The proposed algorithm, called LSHADE44 hereafter, combines competition of strategies from CDE, the use of the archive employed to creating a trial point by a strategy with current-to-pbest/1 mutation, adaptation of F and *CR* parameters with using four pairs of historical circle memories, and linear decreasing of the population size. The string '44' in the label of the algorithm indicates four competing DE strategies, each with its extra pair of historical circle memories.

A. Mechanism of Competition

The competition of strategies proposed by Tvrdík [7], [21] is used to select a strategy when a trial point is created. In this approach, we choose randomly among H different DE strategies at hand with probabilities $q_j,\ j = 1, 2, \ldots, H$.

These probabilities change according to the strategies' success rates in the preceding steps of the search process. The jth strategy is considered successful if it generates a trial point $\boldsymbol{y}$ better than original point $\boldsymbol{x}_i$, i.e. $f(\boldsymbol{y}) \leq f(\boldsymbol{x}_i)$, see line 8 of Algorithm 1. Probability q_j is evaluated as the relative frequency

$$q_j = \frac{n_j + n_0}{\sum_{k=1}^{H}(n_k + n_0)}\ , \tag{1}$$

where n_j is the current count of the jth strategy's successes, and $n_0 > 0$ is a constant. The input parameter $n_0 > 1$ prevents a dramatic change in q_j by one random successful use of the jth strategy. To avoid degeneration of the search process, the current values of q_j are reset to their starting values ($q_j = 1/H$) if any probability q_j decreases below the given limit $\delta > 0$ during the search process.

B. Strategies Selected for LSHADE44

The current-to-pbest/1 mutation (2) was taken from the L-SHADE algorithm [13]. This mutation was proposed originally for JADE algorithm [8] as a modification of the current-to-best/1 mutation.

$$\boldsymbol{u} = \boldsymbol{x}_i + F \cdot (\boldsymbol{x}_{pbest} - \boldsymbol{x}_i) + F \cdot (\boldsymbol{x}_{r1} - \boldsymbol{x}_{r2}). \tag{2}$$

By current-to-pbest/1 mutation (2), the new mutant vector $\boldsymbol{u}$ is generated from four mutually different individuals – the current individual $\boldsymbol{x}_i$, $\boldsymbol{x}_{pbest}$ (which is randomly chosen individual from the $p \times 100\%$ best points of P), randomly selected point $\boldsymbol{x}_{r1}$ from P and randomly selected point $\boldsymbol{x}_{r2}$ from $P \cup A$, A is the archive. In LSHADE44, parameter p is set to a constant.

The randrl/1 mutation was proposed in [22]. It is an improvement of the most popular rand/1 mutation:

$$\boldsymbol{u} = \boldsymbol{r}_1 + F\left(\boldsymbol{r}_2 - \boldsymbol{r}_3\right). \tag{3}$$

Mutation rand/1 computes mutant vector from three points $\boldsymbol{r}_1$, $\boldsymbol{r}_2$, and $\boldsymbol{r}_3$ randomly chosen from current generation of population P. Points $\boldsymbol{r}_1$, $\boldsymbol{r}_2$, and $\boldsymbol{r}_3$ are mutually distinct, and not equal to $\boldsymbol{x}_i$. F is input parameter, so called scaled factor. The randrl/1 mutation (4) is very similar to rand/1, only $\dot{\boldsymbol{r}}_1$ is the tournament best among $\boldsymbol{r}_1$, $\boldsymbol{r}_2$, and $\boldsymbol{r}_3$ and the points $\dot{\boldsymbol{r}}_2$ and $\dot{\boldsymbol{r}}_3$ are the remaining points of randomly selected triplet.

$$\boldsymbol{u} = \dot{\boldsymbol{r}}_1 + F\left(\dot{\boldsymbol{r}}_2 - \dot{\boldsymbol{r}}_3\right). \tag{4}$$

Kaelo and Ali [22] found on the set of 50 difficult benchmark problems that the randrl/1 mutation can increase the speed of the search up to 30 % without significant decreasing the reliability of the search compared to the rand/1 mutation. The conclusions of [23], where also both (3) and (4) mutations are compared, are in agreement with [22].

After the mutation, a crossover is used in generating a new trial point. Either the binomial crossover or the exponential one is employed in LSHADE44 algorithm.

The binomial crossover combines coordinates of $\boldsymbol{x}_i$ with coordinates of mutant $\boldsymbol{u}$ into trial point $\boldsymbol{y}$ according to formula (5).

$$y_j = \begin{cases} u_j & \text{if} \quad U_j \leq \mathit{CR} \quad \text{or} \quad j = l \\ x_{ij} & \text{otherwise,} \end{cases} \tag{5}$$

where l is a number randomly chosen from set $\{1, 2, \ldots, d\}$, $U_1, U_2, \ldots, U_d$ are independent random variables uniformly distributed in $[0,\ 1)$. The *CR* is an input parameter of DE, so called crossover parameter, $\mathit{CR} \in [0, 1]$.

The binomial crossover is employed preferably in the differential evolution, see e.g. [3], [4], [9], [24]. The reasons for the usage of the exponential crossover were found in the results of the papers, where the influence of DE strategies on the performance of the algorithm was studied in various benchmarks [16], [17], [20], [21]. The comparisons show that the use of DE strategies with the exponential crossover often increases the efficiency of the algorithm.

The exponential crossover combines the coordinates of original point $\boldsymbol{x}_i$ and mutant $\boldsymbol{u}$ into trial point $\boldsymbol{y}$ by the following way. L consecutive coordinates are moved into trial point $\boldsymbol{y}$ from mutant $\boldsymbol{u}$. The coordinate u_m, where the m is randomly chosen with uniform distribution from set $\{1, 2, \ldots, d\}$, is always moved into trial point $\boldsymbol{y}$. Next coordinate is moved with probability *CR*, kth member in sequence of L consecutive coordinates of point $\boldsymbol{u}$ moves into point $\boldsymbol{y}$ with probability proportional to CR^{k-1}. Coordinates, which are not copied into trial point $\boldsymbol{y}$ from $\boldsymbol{u}$, are copied from $\boldsymbol{x}_i$.

The *CR* parameter influence the number of elements to be put into $\boldsymbol{y}$ from $\boldsymbol{x}_i$ and from $\boldsymbol{u}$ for both discussed types of crossover. However, the relation between *CR* and the probability of the mutation is different. The mean number of elements coming from the mutant vector into the trial vector is proportional to the probability of the mutation.

Let p_m be a mutation probability for a coordinate of $\boldsymbol{x}_i$ by respective coordinate of mutant $\boldsymbol{u}$. Relations between the probability p_m and *CR* for both discussed types of crossover were studied and derived by Zaharie [25]. The relation between the probability p_m and *CR* is linear for binomial crossover, the relation for exponential crossover is not linear, described by polynomial (6).

$$\mathit{CR}^d - \ d\ p_m\ \mathit{CR} + \ d\ p_m - 1 = 0. \tag{6}$$

The relation (6) has to be taken into account when we set up the proper value of *CR* for the exponential crossover.

When we combine the two kinds of mutation and two types of crossover mentioned in this subsection, we obtain four DE strategies competing in LSHADE44 algorithm. They are current-to-pbest/1/bin, current-to-pbest/1/exp, randrl/1/bin, and randrl/1/exp.

C. Archive and its Updating

The archive A of old members of population is also used in our algorithm. The points from archive A are used in the mutation according to (2). If the function value of the newly generated trial point $\boldsymbol{y}$ is better than the function value in $\boldsymbol{x}_i$, the trial point $\boldsymbol{y}$ replaces the $\boldsymbol{x}_i$ in population and the point $\boldsymbol{x}_i$ is inserted into archive A. At the beginning, the archive A is set to $A = \emptyset$. When the size of the archive A exceeds its maximum size K, randomly chosen superfluous points are deleted.

D. Adaptation of F and CR and Historical Circle Memories

The adaptation of F and *CR* proposed for SHADE algorithm is slightly modified due to usage more than one DE strategy. We use four pairs of the historical circle memories (M_F and M_C), a pair for each strategy used in LSHADE44.

When a strategy uses binomial crossover, the M_C memory serves to adaptation of *CR*. When a strategy uses exponential crossover, the M_C memory serves to adaptation of p_m, and then the value of *CR* is evaluated from the value of p_m in accordance with (6).

The evaluation of *CR* according to (6) is time-consuming when it is applied to each use of DE strategy with the exponential crossover. Therefore, the table of $(d_p + 1)$ pairs of p_m and *CR* for transferring p_m into *CR* is pre-computed at the initial stage of the run in our implementation. The interval $[1/d, 1]$ is separated into d_p intervals of the same length. The *CR* is then computed by using (6) for each $p_m = 1/d + r \times (1 - 1/d)/d_p$, $0 \leq r \leq d_p$. Then if we need to transfer p_m^x into CR^x, we find the closest p_m to our current value p_m^x in the table and we take its corresponding *CR* as searched CR^x.

In preliminary experiments, we also tested a version of algorithm with direct adaptation of *CR* values for the exponential crossover but the results achieved by this approach were worse compared to the version adapting the p_m values and their storage in the corresponding M_C memory for the strategies with the exponential crossover.

All the circle memories are of the same size, H_m, and all the elements of each pair of M_F and M_C circle memories are set to 0.5 at the beginning. Each pair of the memories has its pointer (k) to the position being updated. The value of k is set up to one at the beginning of the search and the pointer is increased by 1 after each update. The memories are updated in circular manner, i.e. if the increased pointer is higher than H_m, it is set to one.

Algorithm 2 LSHADE44 algorithm

```
initialization: N_init, N_min, circle memories, probabilities q_j = 1/4, j = 1,2,3,4, archive A = ∅
N := N_init
generate an initial population P = (x_1, x_2, ..., x_N)
evaluate f(x_i), i = 1, 2, ..., N
while stopping condition not reached do
    set all S_F and S_C empty; Q = ∅
    for i = 1 to N do
        choose strategy z according to the current probabilities q_j
        generate F and CR, use respective circle memories
        generate a trial vector y
        evaluate f(y)
        if f(y) < f(x_i) then
            save F and CR (or p_m) into respective S_F and S_C
            insert x_i into archive A
        end if
        if f(y) ≤ f(x_i) then
            insert y into new generation Q
            increase n_z
        else
            insert x_i into new generation Q
        end if
    end for
    P := Q
    modify circle memories if needed, use respective S_F and S_C for each pair
    N_old := N, recompute size of population N, eq. (13)
    if N < N_old then
        remove superfluous points from population
    end if
end while
```

The elements of M_F and M_C memories are updated after each generation. Precisely, the actual (pointed) pair of members of M_F and M_C is updated, when the respective strategy has generated at least one successful trial point ($f(\boldsymbol{y}) < f(\boldsymbol{x}_i)$) in the just completed generation. When the trial point $\boldsymbol{y}$ is successful, the F and *CR* (for the strategies with the binomial crossover) and F and p_m (for the strategies with the exponential crossover) are stored into the respective S_F and S_C sets. All the S_F and S_C sets are set as empty set at the beginning of each generation. After generation, a pair of S_F and S_C is used for the modification of the actual (pointed by k) member of the M_F and M_C memories with respect to the DE strategy. The modification is carried out as follows. The new value of M_{F_k} is computed as weighted Lehmer mean of the current values from S_F according to (7) and (9) and the value of M_{C_k} is computed as weighted arithmetic mean of the current values from S_C according to (8) and (10). The means are weighted by the difference between function value in $\boldsymbol{y}_s$ and $\boldsymbol{x}_s$, which was replaced by $\boldsymbol{y}_s$ (see equations (11) and (12)). The values of M_{F_k}, M_{C_k}, and the pointer k remain the same if no successful point was created by respective strategy in the last generation.

$$M_{F_k} = \mathit{mean}_{WL}(S_F) \quad \mathit{if} \quad S_F \neq \emptyset, \tag{7}$$

$$M_{C_k} = \mathit{mean}_{WA}(S_C) \quad \mathit{if} \quad S_C \neq \emptyset, \tag{8}$$

$$\mathit{mean}_{WL}(S_F) = \frac{\sum_{s=1}^{|S_F|} w_s \; F_s^2}{\sum_{s=1}^{|S_F|} w_s \; F_s}, \tag{9}$$

$$\mathit{mean}_{WA}(S_C) = \sum_{s=1}^{|S_C|} w_s \; V_s, \tag{10}$$

$$w_s = \frac{\Delta f_s}{\sum_{h=1}^{|S_C|} \Delta f_h}, \tag{11}$$

$$\Delta f_s = |f(\boldsymbol{x}_s) - f(\boldsymbol{y}_s)|, \tag{12}$$

where V_s is either CR_s or p_{m_s}.

The values of F and *CR* parameters (*CR* via p_m in the strategies using the exponential crossover) are generated before each computing of a new trial point $\boldsymbol{y}$ as follows. For the current DE strategy, a pair of elements from the corresponding M_F and M_C memories is selected randomly and this pair of values is used as the parameters of the location for the distributions from that the values of F and *CR* (or p_m) for the current trial point are generated. Let us denote this randomly selected pair of values by μ_F and μ_C.

The value of F is generated from Cauchy distribution with parameters of $(\mu_F, 0.1)$. If F is less than 0, F is generated again and if F is higher than 1 then it is set to $F = 1$.

The value of *CR* for the strategy with the binomial crossover is generated from normal distribution $N(\mu_C, 0.1)$. If *CR* is less than 0, then it is set to $\mathit{CR} = 0$, if *CR* is higher than 1 then it is set to $\mathit{CR} = 1$. For the strategies with the exponential crossover, the value of p_m is generated from normal distribution with parameters of $(\mu_C, 0.1)$ and the value of *CR* is evaluated using the pre-computed table with the values of p_m and the values of *CR* corresponding to (6). If p_m is less than $1/d$ then $p_m = 1/d$, if p_m is higher than 1 then $p_m = 1$.

E. Decreasing Size of Population

Linearly decreasing of population size was proposed for L-SHADE algorithm [13]. This mechanism is used also in our algorithm. The population size is decreasing linearly generation by generation with the increasing number of the objective function evaluations (*FES*) during the search process from the initial value N_{init} to the final value N_{min} at the end of the search process, i.e. if allowed number of the function evaluations (*MaxFES*) is reached. After each generation, new size of population N is computed.

$$N = \mathit{round}\left[N_{init} - \frac{\mathit{FES}}{\mathit{MaxFES}} \; (N_{init} - N_{min})\right], \tag{13}$$

where *FES* is the current number of the objective function evaluations. Whenever $N < N_{old}$, the $(N_{old} - N)$ worst individuals (i.e. the individuals with the highest function values) are deleted from the population.

TABLE I. VALUES OF FUNCTION ERRORS AND TUNED PARAMETER OF LSHADE44 ALGORITHM, $d = 10$

f	Best	Worst	Median	Mean	Std	N_{init}/d
1	0	0	0	0	0	18
2	0	0	0	0	0	18
3	1.81368	20.0176	20.0064	17.6140	6.056	9
4	1.00347	4.98957	3.00448	3.17729	0.986	36
5	5.47844	103.015	17.9781	22.5679	17.76	36
6	0	11.3811	0.99496	1.34829	1.780	18
7	0.04178	0.66360	0.20624	0.22223	0.127	18
8	0.00026	4.08903	0.31460	0.48014	0.839	36
9	100	100.047	100	100.002	0.007	18
10	140.701	152.289	141.525	142.650	2.634	36
11	1.79730	4.02160	2.93008	2.90081	0.594	36
12	109.500	112.650	111.480	111.481	0.727	36
13	0.09254	0.10275	0.09273	0.09291	0.001	36
14	100	6677.01	6670.66	6542.22	920.1	36
15	100	100	100	100	0	18

TABLE II. VALUES OF FUNCTION ERRORS AND TUNED PARAMETER OF LSHADE44 ALGORITHM, $d = 30$

f	Best	Worst	Median	Mean	Std	N_{init}/d
1	0	0	0	0	0	18
2	0	0	0	0	0	18
3	20.0467	20.1268	20.0882	20.0872	0.018	9
4	18.2046	32.1866	26.2190	25.5617	3.663	18
5	837.806	1724.33	1330.79	1355.23	200.0	9
6	17.6094	382.136	61.6418	105.350	88.25	36
7	2.88568	7.28623	6.37645	6.15153	0.946	9
8	7.45969	61.5395	23.3074	25.7729	12.23	36
9	104.378	106.405	105.935	105.781	0.472	36
10	533.963	643.090	538.620	559.131	35.45	36
11	300.322	652.452	518.755	506.357	107.7	9
12	107.128	109.848	108.435	108.401	0.623	18
13	0.01027	0.01127	0.01063	0.01062	2.1E-04	9
14	33760	43477.3	42595.8	40552.19	3805	18
15	100	100	100	100	0	18

TABLE III. VALUES OF FUNCTION ERRORS AND TUNED PARAMETER OF LSHADE44 ALGORITHM, $d = 50$

f	Best	Worst	Median	Mean	Std	N_{init}/d
1	0.46751	1583.68	31.1971	160.073	333.3	36
2	0	0	0	0	0	18
3	20.1421	20.2407	20.1832	20.1843	0.021	9
4	42.0267	64.9766	51.1007	52.5590	4.920	9
5	2057.82	3350.82	2868.62	2842.29	263.7	9
6	419.695	1814.48	995.850	1027.51	296.8	36
7	38.2080	41.4513	40.2962	40.2554	0.622	18
8	18.8285	594.262	147.185	189.023	150.8	36
9	102.144	103.030	102.630	102.607	0.211	36
10	613.981	1497.41	916.051	955.535	208.2	18
11	400	457.409	400.001	412.217	13.92	36
12	114.428	201.536	115.343	120.433	20.48	36
13	0.02453	0.02575	0.02512	0.02517	3.0E-04	9
14	52657.3	52706.6	52678.2	52673.28	13.69	9
15	100	100	100	100	0	18

The new LSHADE44 algorithm with competing strategies is described by pseudo-code in Algorithm 2.

IV. EXPERIMENTS

All computations were carried out on a standard PC with Windows 7, Intel(R) Core(TM)i7-4600U CPU, 2.10GHz, 2.70GHz, 8 GB RAM.

The LSHADE44 algorithm described above is implemented in Matlab R2015b and this environment was used for experiments. Experimental setting follows the requirements given in the report [1]. Fifteen minimization problems are defined in the report, the source code of functions in C cec15_func.cpp was downloaded from the web page [1] and compiled via mex command (mex cec15_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 $d = 100$.

V. RESULTS

The report [1] stated the obligatory content and structure of the experimental results, which is presented below.

A. Function errors

The values of function errors are shown in Tables I, II, III, and IV for $d = 10$, $d = 30$, $d = 50$, and $d = 100$, respectively. Due to the opportunity of parameters tuning in Learning-based case part of Special Session and Competitions on Real-Parameter Single Objective Optimization on CEC2016 to which the algorithm is applied, we tested the algorithm with three different values of N_{init}. The parameter is the size of population at the beginning of the search. According to our experience, this parameter can affect founded solution highly. The three used settings were $N_{init} = 9 \times d$, $N_{init} = 18 \times d$, and $N_{init} = 36 \times d$. The value of $N_{init} = 18 \times d$ is recommended by authors of L-SHADE algorithm [13]. Thus, we tested the LSHADE44 algorithm with smaller than recommended, with recommended, and with higher than recommended size of population at the beginning of the search. For each problem and each dimension, the results of the best of these three tested settings are displayed in the Tables I, II, III, and IV. The global minimum point and the function value in it are known for each problem, thus the function errors could be computed and are depicted in the tables. The values of function error smaller than 1×10^{-8} were replaced by zeros. The tables are presented in the structure required by [1], only the column labelled N_{init}/d is added to identify the best setting. In all cases, where more than one winner were, the recommended setting was beside them, so we placed the originally recommended constant in the column N_{init}/d. Unfortunately, the values in the columns N_{init}/d in the tables do not give us any clear instruction how to choose the appropriate value for the parameter N_{init}.

The performance of LSHADE44 and original L-SHADE on the CEC 2015 benchmark problems was compared statistically by Wilcoxon two-sample test. The significance level for these statistical tests was 0.05. The results of the comparison are depicted in Table VI. The sign "+" means that LSHADE44

TABLE IV. VALUES OF FUNCTION ERRORS AND TUNED PARAMETER OF LSHADE44 ALGORITHM, $d = 100$

f	Best	Worst	Median	Mean	Std	N_{init}/d
1	57181	248070	118166	125054	3.9E+04	36
2	0	0	0	0	0	18
3	20.3698	20.4889	20.4443	20.4417	0.027	9
4	81.9654	142.279	105.744	107.083	10.76	9
5	8311.47	10200.2	9617.18	9590.42	383.4	9
6	3130.47	6202.33	4823.9	4778.16	703.9	36
7	93.4236	105.684	97.0074	97.6626	2.893	18
8	841.341	3417.53	2096.35	2153.75	556.2	18
9	107.848	111.720	108.908	109.313	1.189	18
10	2739.95	4879.08	3503	3533.47	483.6	36
11	402.087	580.325	468.248	471.429	39.69	36
12	112.278	113.894	113.499	113.437	0.341	36
13	0.05981	0.06594	0.06315	0.06308	0.001	9
14	108828	108899	108869	108867	17.32	18
15	100	100	100	100	0	18

TABLE V. COMPUTATIONAL COMPLEXITY

d	*T0 (s)*	*T1* (s)	$\hat{T2}$ (s)	$(\hat{T2} - T1)/T0$
10	0.2393	0.58	17.37	70.16
30	0.2393	1.39	34.17	136.98
50	0.2393	2.3	73.49	297.49
100	0.2393	5.37	367.10	1511.62

was significantly better than L-SHADE , the sign "−" means that LSHADE44 was significantly worse than L-SHADE, and the sign "=" is used for no significant difference between the performance of both algorithms. It is apparent from the table that the proposed LSHADE44 algorithm performs appreciably better than original algorithm on the benchmark set defined for competition on CEC2015.

B. Algorithm complexity

Algorithm complexity was measured according to the guidelines in [1], the results are presented in Table V, the values of time are measured in seconds. *T0* is the time needed for computing the source code given in [1] carrying out 1×10^6 times of the prescribed sequence of arithmetic operations, *T1* is the time of 2×10^5 evaluations of Function 1 and $\hat{T2}$ is the average time of five runs of the algorithm with *MaxFES* $= 2 \times 10^5$ on Function 1.

TABLE VI. STATISTICAL COMPARISON OF LSHADE44 AND L-SHADE ALGORITHMS, RESULTS OF WILCOXON TWO-SAMPLE TESTS

f/d	10	30	50	100
1	=	=	+	+
2	=	=	=	=
3	=	+	+	+
4	=	=	−	+
5	=	=	+	+
6	+	+	+	+
7	=	+	+	+
8	+	+	+	+
9	−	+	+	+
10	+	+	+	+
11	=	=	=	+
12	+	+	+	−
13	+	+	=	+
14	−	−	+	+
15	=	=	=	=
total# +/-	5/2	8/1	10/1	12/1

C. Parameters

The detailed description of all parameters of proposed algorithm is required in this section by [1]. LSHADE44 algorithm has some parameters. Each of them is joined with one of the incorporated adaptive mechanisms. We undertook the setting for almost all of them from the original algorithm [13], [21]. The only tuned parameter in our algorithm is the size of population at the beginning of the search, N_{init}. We set the parameter to three different levels, the original one $N_{init} = 18 \times d$, to the half of this level, and to the double of the level, these two additional levels seemed rational for us. The results of the best of these three tested settings were submitted for each problem in each dimension.

The auxiliary parameter d_p used in the implementation for determining the number of rows in the pre-computed table converting the p_m into *CR* (see polynomial (6)) was set intuitively to keep the sufficient accuracy of respective *CR* values for the strategies with exponential crossover. The summary of the parameters setting is as follows.

- Competition mechanism:
  $H = 4$, $\delta = 1/(H * 5) = 1/20$, $n_0 = 2$,
- probability p in the current-to-pbest/1 mutation (2):
  $p = 0.11$,
- archive A:
  maximum size of archive A is set to $K = 2.6 \times N$,
- adaptation of F and *CR*:
  $H_m = 6$, $d_p = 100$,
- linearly decreasing the size of population:
  $N_{min} = 4$, N_{init} is set individually for each function in each dimension, see Tables I, II, III, and IV.

VI. CONCLUSIONS

The new adaptive variant of differential evolution algorithm was proposed. The algorithm is enhanced version of L-SHADE algorithm, it is called LSHADE44 algorithm. The presented algorithm is applied to Learning-based case part of Special Session and Competitions on Real-Parameter Single Objective Optimization on CEC2016. From statistical comparison between it and original version L-SHADE, it is clear that the proposed algorithm searches better solution for more than half from used benchmark problems.

The proposed LSHADE44 algorithm is almost control-parameter free and we hope that such type of the algorithms is beneficial for their application to real-world optimization problems. On the other hand, LSHADE44 algorithm is rather complicated, especially in the evaluating of the proper *CR* value corresponding to the desired value of p_m using the pre-computed tables. We perceive this fact and plan to study deeply

also an easier version with straightforward adaptation of *CR* parameter instead of its adaptation through the p_m parameter. However, such version was preliminary tested and seemed to be worse performing.

The LSHADE44 algorithm seems to be efficient but there are some problems in benchmark set, where the algorithm stagnates or converges slowly. These issues which relate to many adaptive versions of differential evolution still remain open.

ACKNOWLEDGMENT

This work was supported by the project LQ1602 IT4Innovations excellence in science and partially supported by University of Ostrava from the project SGS08/UVAFM/2016.

REFERENCES

[1] J. J. Liang, B. Y. Qu, and P. N. Suganthan, "Problem definition and evaluation criteria for the CEC 2015 competition on learning-based real-parameter single objective optimization," 2014. [Online]. Available: http://www.ntu.edu.sg/home/epnsugan/

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

[3] 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.

[4] 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.

[5] 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.

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

[8] 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.

[9] R. Tanabe and A. Fukunaga, "Success-history based parameter adaptation for differential evolution," in *IEEE Congress on Evolutionary Computation 2013 Proceedings*, 2013, pp. 71–78.

[10] ——, "Evaluating the performance of SHADE on CEC 2013 benchmark problems," in *IEEE Congress on Evolutionary Computation 2013 Proceedings*, 2013, pp. 1952–1959.

[11] J. J. Liang, B. Qu, P. N. Suganthan, and A. G. Hernández-Díaz, "Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization," [online] http://www.ntu.edu.sg/home/epnsugan/, 2013.

[12] I. Loshchilov, T. Stuetzle, and T. Liao, "Ranking results of CEC13 special session and competition on real-parameter single objective optimization," 2013. [Online]. Available: http://www3.ntu.edu.sg/home/epnsugan/

[13] R. Tanabe and A. Fukunaga, "Improving the search performance of SHADE using linear population size reduction," in *IEEE Congress on Evolutionary Computation 2014 Proceedings*, 2014, pp. 1658–1665.

[14] 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," 2013. [Online]. Available: http://www.ntu.edu.sg/home/epnsugan/

[15] ——, "Ranking results of CEC14 special session and competition on real-parameter single objective optimization," 2014. [Online]. Available: http://www3.ntu.edu.sg/home/epnsugan/

[16] P. Bujok, "Success-history based differential evolution with adaptation by competing strategies," 2016, manuscript submitted for publication in Swarm and Evolutionary Computation.

[17] 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.

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

[19] J. Tvrdík, "Adaptation in differential evolution: A numerical comparison," *Applied Soft Computing*, vol. 9, no. 3, pp. 1149–1155, 2009.

[20] M. Weber and F. Neri, "Contiguous binomial crossover in differential evolution," in *Lecture Notes in Computer Science 7269*. Springer, 2012, pp. 145–153.

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

[22] 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.

[23] R. Poláková, "A comparison of two similar mutation operators in differential evolution," in *21st International Conference on Soft Computing MENDEL 2015*, 2015, pp. 1–6.

[24] R. Mallipeddi and P. N. Suganthan, "Ensemble of constraint handling techniques," *IEEE Transactions on Evolutionary Computation*, vol. 14, pp. 561–579, 2010.

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