LSHADE Algorithm with Rank-Based Selective Pressure Strategy for Solving CEC 2017 Benchmark Problems

Vladimir Stanovov
Department of System Analysis and Control
Reshetnev Siberian State University of Science and Technology
Krasnoyarsk, Russia
vladimirstanovov@yandex.ru

Shakhnaz Akhmedova
Department of Higher Mathematics
Reshetnev Siberian State University of Science and Technology
Krasnoyarsk, Russia
shahnaz@inbox.ru

Eugene Semenkin
Department of System Analysis and Control
Reshetnev Siberian State University of Science and Technology
Krasnoyarsk, Russia
eugenesemenkin@yandex.ru

***Abstract*—Solving single-objective real-parameter optimization problems can still cause difficulties, for example if the optimized function is multimodal or has rotated trap problems. Such optimization problems can be found in various areas in real-world applications. Usually, these problems are very complex and computationally expensive. A new algorithm, which is a modification of the LSHADE algorithm with a rank-based selective pressure strategy, called LSHADE-RSP, is presented in this paper. The proposed algorithm is a new variant of the LSHADE algorithm, the basic idea of which consists in the adaptation of its mutation strategy using selective pressure. The experiments were performed on CEC 2018 benchmark functions. A comparison of the proposed LSHADE-RSP algorithm and the algorithm-participants of the CEC 2017 competition is presented. From the obtained results it can be concluded that LSHADE-RSP performs better in comparison with most alternative algorithms: using the CEC 2018 evaluation method, LSHADE-RSP obtained one of the best final scores among the algorithms that were winners of the previous competition.**

Keywords—LSHADE, selective pressure, covariance matrix, optimization, crossover, mutation.

I. Introduction

Research on single-objective optimization algorithms is the basis of research on more complex optimization algorithms such as multi-objective optimization algorithms, constrained optimization algorithms, optimization problems with different types of variables and so on. Thus, initially all optimization algorithms should be tested on single-objective benchmark problems to determine their workability.

Nowadays various real-world problems in engineering and related areas can be reduced to optimization problems, which usually have many local optima, so it is difficult to find their global optima. For solving such problems, researchers have presented many methods over recent years and population-based algorithms are among them [1].

Among the variety of population-based algorithms, one of the most effective is the Differential Evolution (DE) algorithm [2]. Recently the LSHADE algorithm, which is a modification of DE techniques, has been proposed [3]. Although the LSHADE algorithm has been applied successfully in solving many difficult optimization problems, there still occur difficulties in keeping the balance between exploration and exploitation when solving complex multimodal problems. In order to achieve better performance, in this study a new version of the considered population-based algorithm was proposed.

In this paper, the problems of premature convergence and search diversification were solved by a modification of the LSHADE technique's mutation and crossover operators. More specifically, a rank-based selective pressure strategy [4] was used for its mutation strategy. The developed technique was called the "LSHADE Algorithm with Rank-Based Selective Pressure Strategy" or LSHADE-RSP. The efficiency of LSHADE-RSP was examined on test problems taken from the CEC 2018 competition on real-parameter single-objective optimization [5]. Experimental results demonstrated that LSHADE-RSP performs better in comparison with the alternative algorithms.

Therefore, in this paper firstly a brief description of the DE algorithm and consequently its modification LSHADE is given. Then the proposed LSHADE-RSP technique and its parameter settings and adaptation are presented. In the next section, the experimental results obtained by the new developed LSHADE-RSP algorithm are demonstrated and discussed. In addition, the performance of the LSHADE-RSP algorithm was compared to the other methods, which took part in the CEC 2017 competition on single-objective bound constrained optimization. Finally, some conclusions are given in the last section.

II. LSHADE Algorithm

A. Differential Evolution

Differential evolution (DE) is the continuous optimization technique first introduced by K. Price and R. Storn in 1997 [2]. Over the last 20 years, it has become one of the most popular and often prize-winning optimization techniques due to its simplicity in implementation and several important features. As with other biology-inspired methods, DE is a population-based algorithm, and the population contains a number of solutions. One of the main features of DE is the mutation scheme, which has been shown to automatically adapt to the scale of the optimized function, improving the performance.

Research is performed with the support of the Ministry of Education and Science of Russian Federation within State Assignment project №2.1680.2017/ПЧ.

978-1-5090-6017-7/18/$31.00 ©2018 IEEE

The algorithm also contains crossover and selection schemes, which will be briefly described.

The key idea of differential evolution is in constructing a mutant vector using the difference between two other vectors from the current population. There are several popular mutations schemes, each having different behaviour, namely DE/rand/1, DE/rand/2, DE/best/1, DE/best/2, DE/target-to-best/1 (DE/current-to-best/1) [6] and many others. One of the most effective and commonly used mutation schemes is DE/current-to-pbest/1 [7]. The equation for constructing the mutant vector is presented in the following formula:

$$v_j = x_{i,j} + F(x_{b,j} - x_{i,j}) + F(x_{r1,j} - x_{r2,j}). \quad (1)$$

In (1) $x_{i,j}$ is the j-th coordinate of the i-th individual x_i, $r1$ and $r2$ are mutually random numbers representing indexes of the individuals, v_j is the so-called mutant vector, which will be used in crossover operation, and x_b is randomly chosen as one of the top $100p\%$ individuals of the current population with p from the range (0, 1]. The scaling factor F is the parameter, usually in range [0, 1].

The next step is the crossover, which is performed for each individual x_i as a calculation of the trial vector t with the crossover rate Cr. The j-th variable of the trial vector t is the same as the j-th variable of the mutant vector v if a randomly generated number in the range (0, 1) is smaller than the crossover rate Cr, otherwise it is the same as the corresponding variable of the individual x_i. Cr is the control parameter of the algorithm, usually in the range [0, 1]. $Cr = 1$ means that there is no crossover, and the trial vector is equal to the mutant vector.

The selection step is performed after calculating the fitness value of the trial vector. If the trial vector is better than the i-th individual in the population, then it is replaced by the trial vector.

B. LSHADE Algorithm

LSHADE stands for Linear Population Size Reduction (LPSR) Success History based Adaptive (SHA) Differential Evolution [3] and is an extension of the SHADE algorithm [8], based on one of the adaptive DE modifications JADE [7]. LSHADE was first presented at CEC 2014, and ranked as the winner-algorithm for bound-constrained continuous optimization.

As with many other optimization algorithms and according to the CEC 2018 competition, the population of N solutions is generated within the predefined search range according to the following Equation (2):

$$x_{i,j} = rand(x_{L,j}, x_{U,j}), \quad (2)$$

where the function $rand(a, b)$ returns uniform random numbers from the range $[a, b]$, $x_{i,j}$ is the j-th coordinate of the i-th individual in the population, $x_{L,j}$ and $x_{U,j}$ are the lower and the upper bounds for the j-th variable respectively. Also j changes from 1 to D, which determines the dimension of the problem, while $i = 1, \ldots, N$.

The original LSHADE algorithm uses the DE/current-to-pbest/1 mutation scheme, shown in (1). This mutation scheme is capable of a relatively greedy search due to its first part, but also capable of exploring the search space thanks to the second part. The random index $r2$ is uniformly selected from the joint set of the population and the external archive. The external archive keeps parent individuals which were replaced by the new solutions.

The crossover operation of the LSHADE algorithm is the classical binomial crossover, which is similar to the one described for the DE algorithm. The only difference consists in the fact that the j-th variable of the trial vector t is the same as the j-th variable of the mutant vector v if a randomly generated number in the range (0, 1) is smaller than the crossover rate Cr or if j is equal to $jrand$, where $jrand$ is a randomly chosen index ($jrand = 1, \ldots, D$).

The LSHADE algorithm uses the Linear Population Size Reduction (LPSR) scheme [3], which significantly boosts its performance. This scheme decreases the number of individuals in the population by deleting least fit ones at every generation; the new population size is calculated by the following formula:

$$N_{g+1} = round((N_{min} - N_{max}) / NFE_{max} \cdot NFE + N_{max}), \quad (3)$$

where $N_{min} = 4$ is the minimal number of individuals in the population, N_{max} is the initial population size, NFE is the current number of function evaluations, NFE_{max} is the maximal number of function evaluations, and g is the generation number.

III. Proposed Algorithm

A. LSHADE-RSP

In this section, the modification of the LSHADE algorithm with Rank-based mutation (LSAHDE-RSP) is described. The rank-based mutation scheme (*current-to-pbest/r*) modifies the current-to-pbest/1 strategy so that the second part containing two random vectors receives selective pressure. It has been previously proposed for DE in [9]. More precisely, $r1$ and $r2$ are selected according to the rank selection typically used in genetic algorithms [4], with the ranks assigned as follows:

$$Rank_i = k \cdot (N - i) + 1, \quad (4)$$

with the largest rank assigned to the individual with the largest fitness, and the smallest rank – to the least fit, i.e. here i taken from the range $[1, N]$ is the index in a sorted fitness array. In this study, minimization problems were considered, so a larger fitness means a smaller goal function value. Here k is the scaling factor responsible for the greediness of the rank selection. Thus the probability that the individual i will be selected is calculated as follows:

$$pr_i = Rank_i / (Rank_1 + Rank_2 + \ldots + Rank_N). \quad (5)$$

The new mutation strategy, *current-to-pbest/r*, tends to select individuals with larger fitness values more often, although even the worst individual still has the possibility of being selected for mutation operation. The motivation behind this is that rank-based selection should boost the exploitation capabilities of the mutation strategy without significantly affecting the exploration. The resulting mutation strategy uses the modification proposed for the jSO algorithm, which is an improved modification of LSHADE [10], i.e. different scaling factors for the first and the second part of the equation.

$$v_j = x_{i,j} + Fw(x_{b,j} - x_{i,j}) + F(x_{pr1,j} - x_{pr2,j}), \quad (6)$$

In (6) the Fw parameter takes the following values:

- Fw is equal to $0.7F$ if NFE is smaller than $0.2NFE_{max}$;
- Fw is equal to $0.8F$ if NFE is greater than $0.2NFE_{max}$ (or equal to that value) and smaller than $0.4NFE_{max}$;
- $Fw = 1.2F$ otherwise.

The random indexes *pr1* and *pr2* are chosen according to the pr_i values, calculated using the individual ranks in (5), while the b index is chosen as one of the top $100p$%, the same as in (1). The crossover operation in LSHADE-RSP, as well as many other parameter settings, is taken from the jSO algorithm [10], as it appeared to be one of the best pure-DE optimization techniques at the CEC'17 competition. According to jSO, the Cr values are modified as follows:

- Cr is equal to 0.7 if the newly generated $Cr < 0.7$ and NFE is smaller than $0.25NFE_{max}$
- Cr is equal to 0.6 if the newly generated $Cr < 0.6$ and NFE is smaller than $0.5NFE_{max}$

B. Parameter Settings and Adaptation

The two parameters adapted in LSHADE-RSP are the scaling factor F and the crossover rate Cr. The adaptation uses the same scheme as the original LSHADE algorithm, but the initial values and some constraints are taken from [10]. The scaling factor F for every mutation operation is computed using a Cauchy distribution, and Cr is computed using normal distribution as follows:

$$F = randc(\mu F_r, 0.1), \quad (7)$$

$$Cr = randn(\mu Cr_r, 0.1), \quad (8)$$

where μF_r and μCr_r are randomly chosen from the memory M of successful parameter settings (the memory size is defined as H), and r is a uniformly chosen random index. Initially all μF_r are set to 0.3, and μCr_r are set to 0.8, and in addition to this, one memory cell always keeps μF_r and μCr_r, which are equal to 0.9. The values in μF_r and μCr_r in one memory cell are updated at the end of each generation using the Lehmer mean, which takes into consideration the fitness improvement. During the memory update, the new values are calculated as the mean of the old F or Cr value and the newly generated value.

```
Initialize population P_0 = (x_{0,j}, …, x_{N,j}), generation g = 0
Set archive A as an empty set
Set μF = 0.3, μCr = 0.8, NFE = 0
While NFE < NFE_max
  S_F and S_Cr are empty sets at the beginning
  For i = 1, …, N
    Randomly select memory index r from [1, H]
    If r = H
      F = randc(0.9, 0.1)
      Cr = randn(0.9, 0.1)
    Else
      F = randc(μF_r, 0.1)
      Cr = randn(μCr_r, 0.1)
    End if
    If μCr_r < 0
      Cr = 0
    End if
    If NFE < 0.25NFE_max
      Cr = max(μCr_r, 0.7)
    Else if NFE < 0.5NFE_max
      Cr = max(μCr_r, 0.6)
    End if
    Generate mutant vector v with current-to-pbest/r
    Apply binomial crossover to v to get trial vector t
    If f(t) < f(x_i)
      Put x_i into archive set A
      Put Cr into S_Cr
      Put F into S_F
    End if
    Update population size by removing worst solutions
    Update archive size by removing random solutions
    Update μF and μCr
  End for
  Update generation number g = g + 1
End while
```

Fig. 1. Pseudo-code of the LSHADE-RSP algorithm

For the first $0.6NFE_{max}$ evaluations the F value is constrained to be not larger than 0.7 and not larger than 1.0 during the remaining computational resource. The pb value for the *current-to-pbest/r* strategy, responsible for the greediness of the search, is computed by the following formula:

$$pb = 0.085 + 0.085NFE/NFE_{max}. \quad (9)$$

This setting is similar to the one used in [10]. The idea behind increasing the number of best individuals is to prevent premature convergence by gradually decreasing selective pressure as the algorithm runs. While we use this technique proposed in jSO, the rank calculation for the second part of the equation (6) is not changed.

The pseudo-code of the proposed LSHADE-RSP algorithm is presented in the Fig.1.

IV. Experimental Results

A. Benchmarks and Parameters

The performance of the LSHADE-RSP algorithm was evaluated on the CEC 2018 Competition on Single-Objective Real Parameter Numerical Optimization [5]. The benchmark contains 30 test functions with various features such as a large number of local optima, asymmetry, nonseparability and so on. In addition, all functions are shifted and rotated. The functions in the competition are tested for the corresponding numbers of variables: 10 (10*D*), 30 (30*D*), 50 (50*D*) and 100 (100*D*).

In this study, the following values were chosen for the rest of the algorithm's parameters:

- population sizes $N_{min} = 4$ and $N_{max} = 75 \cdot D^{(2/3)}$;
- memory size $H = 5$;
- rank greediness factor k in (4) was set to 3;
- archive size is equal to N, it is dynamically adjusted;
- 10,000*D* function evaluations, where *D* is the number of variables;
- number of independent runs for every function is equal to 51.

Additional experiments were performed with $k = 0$, i.e. with the same set of parameters, but with identical ranks, resulting in equal probabilities and zero selective pressure.

The experiments were performed using the following system: CPU: Intel® Core™ i5-2410M 2.3GHz, memory 8GB DDR3, Programming language: C++, compiler: g++ (GNU GCC).

B. Algorithm Complexity

The algorithm complexity is estimated by calculating *T*0, *T*1 and *T*2 values [5]. *T*0 is estimated by evaluating mathematical functions, *T*1 – by performing 200000 calculations of the 18-th function f_{18} for all dimensions, and *T*2 by averaging 5 runs of the algorithm on f_{18} with 200000 evaluations. Table I contains the measured values for 10, 30 and 50 variables.

TABLE I. Algorithm Complexity

D	*T*0	*T*1	*T*2	(*T*2–*T*1) / *T*0
10	0.382	0.361	3.2064	7.44869
30		1.007	7.681	17.4712
50		2.212	12.6452	27.312

C. Algorithm Performance

During the algorithm run the error was calculated as the difference between the current best solution $f(x)$ and the global optimum $f(x^*)$. If this difference was less than 10^{-8}, then it was considered to be small enough and taken as zero. Tables II-V contain the worst, best, median, mean and standard deviation values for every function calculated over 51 program runs for 10, 30, 50 and 100 variables respectively.

TABLE II. Algorithm results for 10*D*

№	Worst	Best	Median	Mean	Std. Dev
f_1	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
f_2	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
f_3	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
f_4	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
f_5	3.9798e+00	0.0000e+00	1.9899e+00	1.5607e+00	8.1845e-01
f_6	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
f_7	1.3810e+01	1.0436e+01	1.1592e+01	1.1691e+01	6.1099e-01
f_8	2.9849e+00	0.0000e+00	1.9899e+00	1.6583e+00	7.7987e-01
f_9	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
f_{10}	1.3835e+02	1.8736e-01	6.8299e+00	2.0923e+01	3.9114e+01
f_{11}	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
f_{12}	6.2443e-01	0.0000e+00	4.1629e-01	3.3874e-01	1.6467e-01
f_{13}	5.5841e+00	0.0000e+00	4.8371e+00	3.1352e+00	2.3021e+00
f_{14}	9.9496e-01	0.0000e+00	0.0000e+00	3.9018e-02	1.9313e-01
f_{15}	5.0000e-01	7.1386e-05	1.2016e-01	2.1627e-01	2.1518e-01
f_{16}	1.1401e+00	2.4195e-02	6.1839e-01	5.6949e-01	2.8840e-01
f_{17}	1.7065e+00	2.0050e-02	3.9941e-01	4.5211e-01	3.3488e-01
f_{18}	5.0000e-01	7.4825e-05	7.5032e-02	1.9916e-01	2.1858e-01
f_{19}	3.9242e-02	0.0000e+00	1.9432e-02	1.1450e-02	1.1702e-02
f_{20}	6.2435e-01	0.0000e+00	3.1217e-01	3.9175e-01	1.4943e-01
f_{21}	2.0505e+02	1.0000e+02	1.0000e+02	1.5035e+02	5.1359e+01
f_{22}	1.0000e+02	1.0000e+02	1.0000e+02	1.0000e+02	0.0000e+00
f_{23}	3.0443e+02	3.0000e+02	3.0000e+02	3.0104e+02	1.5040e+00
f_{24}	3.3170e+02	1.0000e+02	3.2954e+02	3.0563e+02	6.8049e+01
f_{25}	4.4338e+02	3.9774e+02	3.9801e+02	4.0240e+02	1.3500e+01
f_{26}	3.0000e+02	3.0000e+02	3.0000e+02	3.0000e+02	0.0000e+00
f_{27}	3.8952e+02	3.8901e+02	3.8952e+02	3.8936e+02	2.3757e-01
f_{28}	6.1182e+02	3.0000e+02	3.0000e+02	3.1168e+02	5.7868e+01
f_{29}	2.4031e+02	2.2793e+02	2.3379e+02	2.3379e+02	3.1144e+00
f_{30}	3.9469e+02	3.9450e+02	3.9450e+02	3.9451e+02	3.9840e-02

TABLE III. Algorithm results for 30*D*

№	Worst	Best	Median	Mean	Std. Dev
f_1	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
f_2	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
f_3	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
f_4	6.4117e+01	5.8562e+01	5.8562e+01	5.8779e+01	1.0784e+00
f_5	1.1081e+01	1.9953e+00	7.9598e+00	7.5953e+00	2.0252e+00
f_6	1.3687e-07	0.0000e+00	0.0000e+00	2.6838e-09	1.8977e-08
f_7	4.1471e+01	3.4885e+01	3.7702e+01	3.7959e+01	1.7138e+00
f_8	1.1941e+01	1.9931e+00	7.9630e+00	7.5332e+00	1.8138e+00
f_9	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
f_{10}	1.9476e+03	5.8946e+02	1.4594e+03	1.4688e+03	3.0108e+02
f_{11}	6.1973e+01	0.0000e+00	9.9497e-01	3.0526e+00	8.5462e+00
f_{12}	3.6304e+02	3.5147e-01	1.2603e+02	1.0993e+02	9.3723e+01
f_{13}	2.3368e+01	5.1377e-01	1.7013e+01	1.6004e+01	4.5620e+00
f_{14}	2.4738e+01	2.0055e+01	2.1241e+01	2.1664e+01	1.2097e+00
f_{15}	3.0159e+00	3.3024e-01	5.0492e-01	7.2992e-01	5.3154e-01
f_{16}	2.5413e+02	4.5134e+00	1.7726e+01	3.4714e+01	4.8990e+01
f_{17}	4.4031e+01	1.2050e+01	3.3187e+01	3.1535e+01	6.8728e+00
f_{18}	2.2196e+01	5.1081e-01	2.0691e+01	2.0408e+01	2.8367e+00
f_{19}	5.4875e+00	2.1417e+00	3.0221e+00	3.1990e+00	7.9410e-01
f_{20}	4.6935e+01	5.8217e-01	2.8520e+01	2.8843e+01	6.8284e+00
f_{21}	2.1100e+02	2.0432e+02	2.0758e+02	2.0758e+02	1.6596e+00
f_{22}	1.0000e+02	1.0000e+02	1.0000e+02	1.0000e+02	0.0000e+00
f_{23}	3.6029e+02	3.4290e+02	3.5014e+02	3.5038e+02	3.3514e+00
f_{24}	4.3312e+02	4.2208e+02	4.2651e+02	4.2685e+02	2.2322e+00
f_{25}	3.8671e+02	3.8669e+02	3.8669e+02	3.8670e+02	5.5921e-03
f_{26}	1.0576e+03	7.8954e+02	9.2915e+02	9.3489e+02	5.2561e+01
f_{27}	5.0958e+02	4.7821e+02	4.9489e+02	4.9504e+02	6.9738e+00
f_{28}	4.1398e+02	3.0000e+02	3.0000e+02	3.0649e+02	2.6008e+01
f_{29}	4.6162e+02	3.6781e+02	4.3751e+02	4.3721e+02	1.3779e+01
f_{30}	2.1225e+03	1.9417e+03	1.9705e+03	1.9713e+03	2.3049e+01

TABLE IV. ALGORITHM RESULTS FOR 50*D*

№	Worst	Best	Median	Mean	Std. Dev
f_1	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
f_2	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
f_3	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
f_4	1.4231e+02	7.5025e-01	2.8513e+01	5.3021e+01	4.3791e+01
f_5	2.2571e+01	8.9797e+00	1.5303e+01	1.4719e+01	3.4835e+00
f_6	1.1157e-06	0.0000e+00	0.0000e+00	9.5313e-08	2.3057e-07
f_7	7.0886e+01	6.0364e+01	6.6198e+01	6.6105e+01	2.6133e+00
f_8	2.2927e+01	6.2873e+00	1.6078e+01	1.5813e+01	3.0103e+00
f_9	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
f_{10}	3.9332e+03	2.1781e+03	3.4081e+03	3.3420e+03	3.8768e+02
f_{11}	3.1142e+01	1.8265e+01	2.4937e+01	2.4746e+01	3.4078e+00
f_{12}	2.0977e+03	2.7779e+02	1.4202e+03	1.4166e+03	3.7013e+02
f_{13}	4.7983e+01	4.5222e+00	3.3697e+01	2.4945e+01	1.6635e+01
f_{14}	2.8533e+01	2.0109e+01	2.3975e+01	2.3896e+01	1.8814e+00
f_{15}	2.4288e+01	1.8024e+01	2.1094e+01	2.0986e+01	1.7036e+00
f_{16}	6.9185e+02	1.3154e+02	3.4087e+02	3.6707e+02	1.6159e+02
f_{17}	4.5416e+02	6.7486e+01	2.5228e+02	2.5469e+02	1.0365e+02
f_{18}	2.6725e+01	2.0602e+01	2.2681e+01	2.2869e+01	1.2354e+00
f_{19}	1.5275e+01	6.7775e+00	1.0977e+01	1.0906e+01	2.0116e+00
f_{20}	3.9711e+02	4.4304e+01	1.0544e+02	1.3990e+02	8.5652e+01
f_{21}	2.2212e+02	2.0745e+02	2.1512e+02	2.1554e+02	3.2013e+00
f_{22}	4.6177e+03	1.0000e+02	2.9973e+03	2.0322e+03	1.8485e+03
f_{23}	4.4335e+02	4.1906e+02	4.3314e+02	4.3264e+02	5.6425e+00
f_{24}	5.1892e+02	5.0153e+02	5.0898e+02	5.0945e+02	3.8999e+00
f_{25}	4.9185e+02	4.8023e+02	4.8024e+02	4.8160e+02	3.7372e+00
f_{26}	1.2444e+03	1.0361e+03	1.1184e+03	1.1282e+03	5.0062e+01
f_{27}	5.8147e+02	4.9498e+02	5.1177e+02	5.1277e+02	1.3609e+01
f_{28}	4.5885e+02	4.5885e+02	4.5885e+02	4.5885e+02	5.6843e-14
f_{29}	3.8981e+02	3.3528e+02	3.6449e+02	3.6420e+02	1.2174e+01
f_{30}	7.3316e+05	5.7941e+05	6.0841e+05	6.1617e+05	3.6372e+04

TABLE V. ALGORITHM RESULTS FOR 100*D*

№	Worst	Best	Median	Mean	Std. Dev
f_1	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
f_2	1.1096e+04	0.0000e+00	0.0000e+00	2.2247e+02	1.5378e+03
f_3	4.3983e-06	0.0000e+00	0.0000e+00	1.4238e-07	6.1302e-07
f_4	2.1581e+02	8.1668e+01	1.9207e+02	1.9333e+02	2.8402e+01
f_5	5.3978e+01	2.6466e+01	4.2040e+01	4.0974e+01	6.1420e+00
f_6	3.1181e-05	2.1073e-07	6.6666e-06	8.2229e-06	5.9581e-06
f_7	1.5564e+02	1.2861e+02	1.4224e+02	1.4209e+02	6.6818e+00
f_8	5.6319e+01	2.4038e+01	3.9734e+01	3.9138e+01	6.8273e+00
f_9	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
f_{10}	1.1443e+04	8.2279e+03	1.0155e+04	1.0089e+04	7.1070e+02
f_{11}	1.5051e+02	4.0563e+01	6.9416e+01	7.7942e+01	2.6098e+01
f_{12}	2.7171e+04	6.0175e+03	1.2957e+04	1.3368e+04	4.9835e+03
f_{13}	1.6320e+02	5.9151e+01	9.3414e+01	9.8763e+01	2.5884e+01
f_{14}	5.6260e+01	3.3264e+01	4.3250e+01	4.3741e+01	6.1145e+00
f_{15}	1.8289e+02	4.6102e+01	9.6763e+01	1.0190e+02	3.0105e+01
f_{16}	2.3083e+03	7.9569e+02	1.7019e+03	1.6757e+03	3.8685e+02
f_{17}	1.6977e+03	6.8873e+02	1.1922e+03	1.1774e+03	2.3313e+02
f_{18}	2.2973e+02	8.3799e+01	1.5552e+02	1.5332e+02	3.5920e+01
f_{19}	8.1673e+01	4.2087e+01	6.0556e+01	6.1673e+01	9.8370e+00
f_{20}	1.9979e+03	9.2326e+02	1.5401e+03	1.5107e+03	2.4900e+02
f_{21}	2.7736e+02	2.4236e+02	2.6322e+02	2.6201e+02	6.5150e+00
f_{22}	1.2010e+04	8.3011e+03	1.0920e+04	1.0854e+04	7.2808e+02
f_{23}	5.8979e+02	5.3848e+02	5.7165e+02	5.7037e+02	1.0926e+01
f_{24}	9.4022e+02	8.9121e+02	9.0850e+02	9.0890e+02	8.7502e+00
f_{25}	7.7769e+02	6.0142e+02	7.0380e+02	7.1033e+02	4.9668e+01
f_{26}	3.4104e+03	3.0643e+03	3.2114e+03	3.2211e+03	7.7578e+01
f_{27}	6.1981e+02	5.4467e+02	5.8380e+02	5.8538e+02	1.7979e+01
f_{28}	5.7686e+02	4.7819e+02	5.2167e+02	5.2310e+02	2.2973e+01
f_{29}	1.8436e+03	8.2234e+02	1.2635e+03	1.2615e+03	2.4343e+02
f_{30}	2.6818e+03	2.0652e+03	2.2695e+03	2.2896e+03	1.3186e+02

The proposed algorithm LSHADE-RSP was able to find the optimal solutions for unimodal functions f_1-f_3 for all dimensions. Among multimodal functions f_4-f_{10} only for the function f_9 has the global optimum been achieved. For hybrid functions f_{11}-f_{20} the 12-th function f_{12} appears to be relatively difficult: the error values for this function are mostly high. For composition functions f_{21}-f_{30} the proposed algorithm appears to get into local optima quite often, for example for f_{22} with 10 and 30 variables the algorithm always achieves the same error value with a standard deviation of 0.

D. Statistical Tests and Comparison to Other Algorithms

The performance of the LSHADE-RSP algorithm was compared to the other methods which took part in the CEC 2017 competition on single-objective bound constrained optimization:

- Effective Butterfly Optimizer using Covariance Matrix Adapted Retreat phase (EBOwithCMAR) [11];
- Single-Objective Real Parameter Optimization: Algorithm jSO [10];
- LSHADE with Semi-Parameter Adaptation Hybrid with CMA-ES (LSHADE-SPACMA) [12];
- Ensemble Sinusoidal Differential Covariance Matrix Adaptation with Euclidean Neighborhood (LSHADE-cnEpSin) [1.].

It was possible due to the fact that test functions were the same for the CEC 2017 and the CEC 2018 competitions. Therefore, all methods had the same amount of computational resources and runs. To compare different methods, the Wilcoxon's rank sum test with $p = 0.05$ was used. Firstly, in Table VI a short comparison between LSHADE-RSP ($k = 3$), LSHADE-RSP ($k = 0$), EBOwithCMAR, jSO and LSHADE-SPACMA is provided.

TABLE VI. COMPARISON BETWEEN LSHADE-RSP ($K = 3$) AND OTHER METHODS

D	EBOwith CMAR	jSO	LSHADE-SPACMA	LSHADE-RSP ($k = 0$)
10	8+/12=/10- 2-	2+/26=/2- 0	12+/14=/4- 8+	2+/25=/3- 1-
30	10+/8=/12- 2-	7+/19=/4- 3+	12+/11=/7- 5+	8+/21=/1- 7+
50	13+/7=/10- 3+	13+/13=/4- 9+	13+/12=/5- 8+	10+/19=/1- 9+
100	13+/8=/9- 4+	16+/9=/5- 11+	8+/6=/16- 8-	15+/15=/0- 15+

Here the numbers of wins, ties and loses of LSHADE-RSP ($k = 3$) over a certain method are shown as the number of pluses, equals and minuses respectively. Under each three numbers, their sum is provided for convenience.

It may be observed that at 10*D* LSHADE-RSP has a similar performance to all methods except LSHADE-SPACMA, so the selective pressure does not give any advantage at this dimension. However, at 30*D*, the difference becomes significant, and LSHADE-RSP with $k = 3$ is better than all other methods, except EBOwithCMAR. Note that here also the significant difference from LSHADE-RSP with $k = 0$ can be

observed, so that it may be concluded that it is the selective pressure that gives an improvement.

At 50*D* the situation is similar, and the advantage of using rank-based selective pressure is more obvious. Moreover, at 50*D* LSHADE-RSP with $k = 3$ is capable of outperforming EBOwithCMAR. Next, at 100*D*, the difference is even larger, but at this dimension LSHADE-SPACMA shows significantly better results, probably due to the hybridisation with CMA-ES.

TABLE VII. COMPARISON FOR 10*D*

№	EBOwith CMAR	jSO	LSHADE-SPACMA	LSHADE-RSP ($k = 3$)
f_1	0.0±0.0 =	0.0±0.0 =	0.0±0.0 =	0.0±0.0
f_2	0.0±0.0 =	0.0±0.0 =	0.0±0.0 =	0.0±0.0
f_3	0.0±0.0 =	0.0±0.0 =	0.0±0.0 =	0.0±0.0
f_4	0.0±0.0 =	0.0±0.0 =	0.0±0.0 =	0.0±0.0
f_5	**0.0±0.0** -	1.7558±0.7526 =	1.7183±0.9045 +	1.4829±0.7974
f_6	0.0±0.0 =	0.0±0.0 =	0.0±0.0 =	0.0±0.0
f_7	**10.5531±0.1731** -	11.7916±0.6008 =	10.9573±0.4104 -	11.7189±0.5072
f_8	**0.0±0.0** -	1.9509±0.7362 +	0.9805±0.6674 =	1.5022±0.9526
f_9	0.0±0.0 =	0.0±0.0 =	0.0±0.0 =	0.0±0.0
f_{10}	37.2101±53.3595 +	35.8974±54.9303 =	**17.8921±33.3181** =	25.0353±47.8957
f_{11}	0.0±0.0 =	0.0±0.0 =	0.0±0.0 =	0.0±0.0
f_{12}	90.1549±73.6373 +	2.6621±16.6166 =	113.7113±61.3424 +	**0.3632±0.1692**
f_{13}	**2.1716±2.5004** -	2.9644±2.3302 =	4.0694±2.4602 =	3.5536±2.2523
f_{14}	0.0605±0.2338 +	0.0585±0.2341 =	0.1546±0.4414 +	**0.039±0.1931**
f_{15}	**0.109±0.1727** =	0.2208±0.1985 =	0.3388±0.1954 +	0.1866±0.2158
f_{16}	**0.417±0.1965** -	0.5688±0.2618 =	0.794±0.3185 +	0.5415±0.2809
f_{17}	**0.1472±0.2008** -	0.5023±0.3446 -	0.1613±0.1596 -	1.0562±2.7807
f_{18}	0.7±2.7403 =	0.308±0.1932 +	2.7686±6.4551 +	**0.2229±0.2122**
f_{19}	0.015±0.0186 +	0.0107±0.0124 =	0.2096±0.3337 +	**0.0085±0.0096**
f_{20}	**0.1469±0.1558** -	0.3428±0.1275 -	0.3072±0.1962 =	0.4346±0.1524
f_{21}	114.0207±34.8462 =	132.3776±47.8889 =	**103.5538±14.4697** =	122.1711±42.2829
f_{22}	**98.4647±10.8565** =	100.0±0.0 =	100.021±0.0876 +	100.0±0.0
f_{23}	300.1747±0.7002 +	301.2056±1.5741 =	301.8258±1.4934 +	**289.1776±58.4413**
f_{24}	**166.2054±98.7333** -	296.5982±78.5413 =	294.3939±82.7522 =	293.0133±83.3642
f_{25}	412.3489±20.9607 +	**405.9629±17.3058** =	419.5348±22.8558 +	410.4304±20.2534
f_{26}	**265.4001±46.9666** -	300.0±0.0 =	300.0±0.0 =	300.0±0.0
f_{27}	391.5744±2.3735 +	**389.3875±0.2231** =	389.4577±0.1651 +	**389.3875±0.2231**
f_{28}	307.1384±71.0908 =	339.0758±95.5958 =	**300.0±0.0** -	318.3425±73.3699
f_{29}	231.1875±3.7306 -	234.196±2.9268 =	**230.9883±3.0402** -	234.1944±2.8874
f_{30}	406.6838±17.6238 +	394.5193±0.0445 =	16444.126±113297.451 +	**394.5096±0.0229**
w/t/l	8+/12=/10-	2+/26=/2-	12+/14=/4-	

Tables VII-X contain the mean and standard deviation of the best solutions of four algorithms for 10, 30, 50 and 100 variables. As can be seen from the tables, LSHADE-RSP ($k = 3$) has the best performance for composition functions f_{11}-f_{20}.

However, the performance on simple multimodal functions has decreased. Yet, as has been shown in Table VI, there are few performance losses when switching from $k = 0$ to $k = 3$, so the performance fall on this function is probably not due to the selective pressure, but some other parameters.

TABLE VIII. COMPARISON FOR 30*D*

№	EBOwith CMAR	jSO	LSHADE-SPACMA	LSHADE-RSP ($k = 3$)
f_1	0.0±0.0 =	0.0±0.0 =	0.0±0.0 =	0.0±0.0
f_2	0.0±0.0 =	0.0±0.0 =	0.0±0.0 =	0.0±0.0
f_3	0.0±0.0 =	0.0±0.0 =	0.0±0.0 =	0.0±0.0
f_4	**56.4521±11.0363** -	58.6705±0.7703 =	58.5616±0.0 -	58.5616±0.0
f_5	**2.7761±1.727** -	8.5568±2.0773 +	3.2652±1.9515 -	7.4282±1.8914
f_6	0.0±0.0 =	0.0±0.0 =	0.0±0.0 =	0.0±0.0
f_7	**33.4609±0.8286** -	38.9268±1.445 =	33.8656±0.8753 -	38.6336±1.905
f_8	**2.0216±1.3043** -	9.0918±1.8218 +	3.2492±1.5171 -	7.5451±2.0598
f_9	0.0±0.0 =	0.0±0.0 =	0.0±0.0 =	0.0±0.0
f_{10}	**1408.0774±212.734** -	1527.6633±274.431 -	1428.9352±227.844 -	1640.3931±293.2794
f_{11}	4.4931±8.6818 +	**3.0375±2.6203** =	18.6656±25.1363 +	4.4658±11.4671
f_{12}	462.8859±260.2657 +	170.3814±100.934 +	582.3874±257.3932 +	**93.9607±66.6525**
f_{13}	14.8926±6.1866 -	14.84±4.7836 -	**13.1162±5.7372** -	17.6226±3.1333
f_{14}	21.8987±3.8012 +	21.8345±1.2336 =	23.0269±1.5124 +	**21.8887±1.3094**
f_{15}	3.6854±2.1308 +	1.0879±0.6845 =	4.7774±2.0535 +	**0.8901±0.6763**
f_{16}	42.6242±56.3761 +	78.923±83.9339 +	30.594±45.2554 =	**18.8199±5.9635**
f_{17}	29.7549±7.4224 -	32.9248±7.9971 =	**29.2732±10.0829** -	35.7103±5.8715
f_{18}	22.1353±1.082 +	20.4111±2.8443 +	23.2888±2.0757 +	**20.0159±3.8568**
f_{19}	8.0404±2.2554 +	4.5031±1.7152 +	9.5641±2.4312 +	**3.3034±0.8887**
f_{20}	35.7234±7.4216 +	**29.3684±5.7971** -	77.9479±52.8941 +	33.0866±6.0162
f_{21}	**198.9054±20.0386** -	209.2889±1.9361 +	208.2431±4.5212 =	207.3378±1.8773
f_{22}	100.0±0.0 =	100.0±0.0 =	100.0±0.0 =	100.0±0.0
f_{23}	351.2183±3.4777 =	**350.7495±3.2667** =	356.0995±2.9802 +	351.0255±2.9835
f_{24}	**418.1112±45.0096** -	426.4564±2.442 =	428.6587±2.3498 +	426.8954±1.8066
f_{25}	**386.532±0.749** -	386.6986±0.0076 =	386.7005±0.0081 +	386.697±0.0067
f_{26}	**537.333±302.8247** -	920.2079±42.53 =	935.2026±50.7721 =	928.5554±41.7234
f_{27}	502.3951±3.9871 +	**497.3851±6.9327** =	506.2149±4.4077 +	497.4917±7.9604
f_{28}	308.3104±28.5162 =	308.7296±29.9517 =	317.2497±40.0442 +	**304.4696±22.1234**
f_{29}	**432.7956±11.2003** -	433.6701±13.507 -	445.2442±12.0637 =	440.8914±11.566
f_{30}	1987.855±41.6975 +	1971.2418±18.7748 =	2001.6784±74.0941 =	**1965.9245±7.9524**
w/t/l	10+/8=/12-	7+/19=/4-	12+/11=/7-	

TABLE IX. COMPARISON FOR 50*D*

№	EBOwith CMAR	jSO	LSHADE-SPACMA	LSHADE-RSP ($k = 3$)
f_1	0.0±0.0 =	0.0±0.0 =	0.0±0.0 =	0.0±0.0
f_2	0.0±0.0 =	0.0±0.0 =	0.0±0.0 =	0.0±0.0
f_3	0.0±0.0 =	0.0±0.0 =	0.0±0.0 =	0.0±0.0
f_4	42.8636±32.9029 -	56.2126±48.2828 =	**30.9175±33.8559** =	61.9651±50.7119
f_5	7.5846±2.3977 -	16.4053±3.4279 +	**6.1648±1.7735** -	13.5405±3.3021
f_6	0.0±0.0 +	0.0±0.0 +	0.0±0.0 -	0.0±0.0
f_7	57.8843±1.5136 -	66.4965±3.4386 =	**56.8666±1.0237** -	67.2027±3.1255
f_8	7.9114±2.4448 -	16.9623±3.1045 +	**5.9112±2.318** -	13.7379±3.5175
f_9	0.0±0.0 =	0.0±0.0 =	0.0±0.0 =	0.0±0.0
f_{10}	**3114.7365±396.775** -	3139.7576±363.545 -	3466.3901±649.252 -	3599.1157±316.59
f_{11}	26.3614±3.3297 +	27.9386±3.2957 +	32.0839±4.4248 +	**24.025±3.5853**
f_{12}	1938.7542±825.678 +	1680.5634±517.772 +	1581.7253±379.378 +	**1372.4842±409.428**
f_{13}	41.4014±24.5301 +	30.5989±21.0166 +	33.7473±16.6787 +	**24.4891±16.8089**
f_{14}	31.2149±3.4827 +	24.9637±1.8549 +	28.7886±2.3452 +	**22.7517±1.5628**
f_{15}	29.3582±5.1452 +	23.8643±2.4637 +	30.9591±4.8489 +	**20.6287±1.7264**
f_{16}	**346.3637±144.3739** =	450.5211±136.3953 +	431.4346±206.1636 =	394.2341±166.1325
f_{17}	274.7809±85.499 =	282.8672±85.2934 =	287.1653±107.7634 =	**254.3916±89.5701**
f_{18}	32.0337±5.9279 +	24.2828±1.9976 +	33.321±7.3736 +	**22.4981±1.1766**
f_{19}	24.4784±3.897 +	14.1386±2.24 +	20.8926±3.3411 +	**10.0993±1.9575**
f_{20}	147.2184±73.7127 =	**140.1016±76.6129** -	178.0237±98.3576 =	158.9622±74.6698
f_{21}	**210.6179±4.017** -	219.1995±3.7284 +	216.246±10.1167 =	215.0429±3.5682
f_{22}	**365.3677±915.2109** -	1487.2365±1735.82 -	1745.126±2007.905 =	2132.9968±1954.67
f_{23}	434.047±8.082 +	**430.0837±6.175** =	442.4666±6.1742 +	431.7619±6.6918
f_{24}	**506.4619±3.8095** -	507.45±4.0866 -	512.7592±6.7431 +	508.9625±3.7407
f_{25}	488.6051±24.4355 +	480.878±2.7722 +	481.5007±3.8178 +	**480.2385±0.0094**
f_{26}	**705.5736±402.3718** -	1128.7789±55.6132 =	1137.4473±58.5184 =	1135.2666±52.4995
f_{27}	522.3751±7.6784 +	**511.2716±10.9673** =	534.4594±12.6334 +	511.3756±11.6254
f_{28}	466.5108±17.7639 +	459.8068±6.7724 =	459.8065±6.7724 +	**458.849±0.0**
f_{29}	**347.3439±19.4973** -	362.9353±13.0271 =	372.9729±32.7126 =	367.204±14.5954
f_{30}	618165.3965±35826.8689 +	**601051.7451±295644.5785** =	642235.9061±60589.6505 +	601494.5686±26398.4208
w/t/l	13+/7=/10-	13+/13=/4-	13+/12=/5-	

TABLE X. COMPARISON FOR 100*D*

№	EBOwith CMAR	jSO	LSHADE-SPACMA	LSHADE-RSP ($k = 3$)
f_1	0.0±0.0 =	0.0±0.0 =	0.0±0.0 =	0.0±0.0
f_2	7.58e+16±5.03e+17 +	8.9403±23.9635 -	1.6896±11.9473 -	**12676.10±89031.56**
f_3	0.0±0.0 +	0.0±0.0 +	0.0±0.0 -	0.0±0.0
f_4	193.3866±30.5955 =	**189.6323±28.6381** =	204.748±9.7708 =	196.8277±12.6706
f_5	28.6587±5.2242 -	43.9078±5.5513 +	**11.998±3.0614** -	36.3058±8.8425
f_6	0.0±0.0 +	0.0002±0.0006 +	0.0±0.0 -	0.0±0.0
f_7	122.1939±4.4226 -	144.8983±6.6369 =	**111.4786±1.4743** -	145.3113±7.2583
f_8	29.6732±7.403 -	42.1525±5.4679 +	**11.3347±3.25** -	36.2944±8.0061
f_9	0.0018±0.0124 =	0.0459±0.1138 +	0.0±0.0 =	0.0±0.0
f_{10}	9905.075±1893.185 -	**9704.3667±674.897** -	9899.9833±914.380 -	10858.439±709.572
f_{11}	65.5798±19.8122 =	113.1717±42.7782 +	**50.1626±33.0559** -	66.7927±23.2887
f_{12}	**4185.0619±781.649** -	18430.044±8268.18 +	4659.7605±822.724 -	11263.657±5121.12
f_{13}	244.6855±87.5059 +	144.8883±37.6307 +	134.3275±37.5648 +	**103.4346±31.7432**
f_{14}	138.3668±29.324 +	64.3406±10.7873 +	73.2415±12.6815 +	**41.4535±5.4588**
f_{15}	165.3581±38.286 +	162.1208±37.6787 +	112.648±37.6072 +	**92.1363±31.6413**
f_{16}	1414.441±372.7498 -	1858.57±345.1195 +	**1271.1687±500.346** -	1646.0752±361.535
f_{17}	1213.6656±254.343 =	1277.9686±236.076 =	**988.016±469.6275** -	1203.6751±241.879
f_{18}	237.1167±58.8036 +	167.2791±36.1391 +	129.2061±35.1811 =	**123.3546±33.1194**
f_{19}	115.3264±18.5904 +	104.8713±19.8807 +	73.7337±11.1757 +	**54.925±5.5638**
f_{20}	1361.9049±306.281 -	1375.9746±240.419 -	**1341.6389±293.614** -	1543.0255±264.626
f_{21}	259.5437±10.466 +	263.7996±6.3652 +	**245.0395±18.1654** -	254.8992±8.7414
f_{22}	10186.64±2672.158 -	10210.468±2160.85 -	**9670.2218±878.480** -	11568.423±780.334
f_{23}	577.3779±13.007 +	571.1545±10.5968 =	581.4312±7.8312 +	**569.7128±10.0637**
f_{24}	918.5495±13.0873 +	**902.1776±7.8154** -	917.3718±22.0409 +	905.781±9.2816
f_{25}	716.1177±36.6878 =	736.0895±34.9543 +	**704.131±40.862** =	720.9613±40.3579
f_{26}	**2774.631±1074.112** =	3267.3082±79.4019 +	3150.0941±73.004 -	3204.4924±93.6678
f_{27}	588.0785±15.1227 +	585.473±21.4584 =	599.5436±20.3548 +	**581.774±15.8283**
f_{28}	**509.7653±59.4968** -	526.8246±27.0362 =	515.0964±19.7943 -	522.8126±21.0501
f_{29}	1279.0472±239.711 =	**1256.5603±189.446** =	1551.9581±288.791 +	1280.1775±197.727
f_{30}	2397.452±149.5904 +	2325.48±117.6139 =	2358.9164±177.223 =	**2305.2406±138.436**
w/t/l	13+/8=/9-	16+/9=/5-	8+/6=16-	

Table XI contains a comparison according to the CEC 2018 evaluation method [5], which consists of the summation of error values and rank-based mean values for each problem for all dimensions. In this comparison, higher weights are assigned to higher dimensions. Note that the function f_2 was excluded from the comparison by the competition organizers due to its numerical instability. The remaining test problems are 1, 3, 4…30.

TABLE XI. ALGORITHMS COMPARISON

D	jSO	LSHADE-SPACMA	LSHADE-cnEpSin	EBOwith CMAR	LSHADE-RSP
10	77.28	34.11	29.32	**100.0**	89.37
30	91.74	75.04	84.38	**100.0**	95.37
50	90.74	76.94	79.64	89.49	**99.91**
100	66.60	**100.0**	96.58	84.52	82.90
Total	88.26	86.57	87.60	93.98	**100.0**

For 10*D* and 30*D,* EBOwithCMAR overcomes LSHADE-RSP. However, the difference is not very large. For 50*D,* LSHADE-RSP is the best method, and for 100*D,* LSHADE-SPACMA has the highest performance. Note that all methods except jSO are hybrid methods, i.e. they are using the CMA-ES algorithm in the implementations, which is known for its good performance at large dimensions, or are using eigenvalue-based crossover, like LSHADE-cnEpSin. In comparison to jSO, which is the only non-hybrid method with classical binomial crossover, LSHADE-RSP shows better results in most cases.

CONCLUSIONS

In this paper, a new population-based algorithm called LSHADE-RSP (LSHADE algorithm with Rank-based mutation) was introduced. It is a modification of the well-known LSHADE algorithm based on the selective pressure for its mutation strategy.

The proposed algorithm was validated and compared with other optimization techniques, namely with participants of the CEC 2017 competition on single-objective real-parameter optimization. Simulations and comparison showed that LSHADE-RSP performs better in comparison with the alternative algorithms. Its workability and effectiveness were established.

Most importantly, LSADE-RSP has proved the importance of the *current-to-pbest/r* mutation strategy, which simply chooses more promising solutions with a larger probability. Thanks to this, the performance of the optimization technique could be significantly increased due to more efficient use of computational resource. Other DE and hybrid methods could be modified to use similar non-uniform probability distribution when selecting random individuals to increase their performance.

The directions of future work may include non-linear rank assignment, developing new mutation strategies and other selection schemes known in the area of evolutionary computation such as proportional selection or tournament selection.

The developed algorithm could be applied to various real-world problems, for example [14], for solving the linear dynamical systems inverse mathematical problem.

REFERENCES

[1] R. Eberhart, Y. Shi, Computational Intelligence: Concepts to Implementations. Morgan Kaufmann, San Francisco, 2007.

[2] R. Storn, K. Price, "Differential Evolution – a Simple and Efficient Heuristic for Global Optimization over Continuous Spaces", Journal of Global Optimization, Vol. 11(4), pp. 341–359, 1997.

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

[4] K. Jebari, M. Madiafi, "Selection Methodsfor Genetic Algorithms", International Journal of Emerging Sciences, Vol. 3(4), pp. 333–344, 2013.

[5] N.H. Awad, M.Z. Ali, J.J. Liang, B.Y. Qu, P.N. Suganthan, "Problem Definitions and Evaluation Criteria for the CEC 2017 Special Session and Competition on Single Objective Bound Constrained Real-Parameter Numerical Optimization", Technical Report, Nanyang Technological University, Singapore, 2016.

[6] S. Das, S.S. Mullick, P.N. Suganthan, "Recent Advances in Differential Evolution – an Updated Survey", Swarm and Evolutionary Computation, Vol. 27, pp. 1–30, 2016.

[7] J. Zhang, A.C. Sanderson, "JADE: Adaptive differential evolution with optional external archive", IEEE Trans. Evol. Comput., Vol. 13, No. 5, pp. 945–958, 2009.

[8] R. Tanabe, A. Fukunaga, "Success-History Based Parameter Adaptation for Differential Evolution", Proceedings of the IEEE Congress on Evolutionary Computation, pp. 71–78, 2013.

[9] W. Gong, Z. Cai, "Differential Evolution With Ranking-Based Mutation Operators", IEEE Transactions on Cybernetics, Vol. 43, Issue 6, pp. 2066-2081, 2013.

[10] J. Brest, M.S. Maucec, B. Boskovic, "Single Objective Real-Parameter Optimization Algorithm jSO", Proceedings of the IEEE Congress on Evolutionary Computation, pp. 1311–1318, 2017.

[11] A. Kumar, R.K. Misra, D. Singh, "Improving the local search capability of Effective Butterfly Optimizer using Covariance Matrix Adapted Retreat Phase", Proceedings of the IEEE Congress on Evolutionary Computation, pp. 1835–1842, 2017.

[12] A.W. Mohamed, A.A. Hadi, A.M. Fattouh, K.M. Jambi, "LSHADE with Semi-Parameter Adaptation Hybrid with CMA-ES for Solving CEC 2017 Benchmark Problems", Proceedings of the IEEE Congress on Evolutionary Computation, pp. 145–152, 2017.

[13] N.H. Awad, M.Z. Ali, P.N. Suganthan, "Ensemble Sinusoidal Differential Covariance Matrix Adaptation with Euclidean Neighborhood for Solving CEC2017 Benchmark Problems", Proceedings of the IEEE Congress on Evolutionary Computation, pp. 372–379, 2017.

[14] I. Ryzhikov, Ch. Brester, E. Semenkin, "A multi-objective approach with a restart meta-heuristic for the linear dynamical systems inverse mathematical problem", International Journal on Information Technologies & Security, № 1 (vol.10), pp. 93-102, 2018