Ranked Archive Differential Evolution with Selective Pressure for CEC 2020 Numerical Optimization*

1st Vladimir Stanovov
Institute of Informatics and Telecommunication
Reshetnev Siberian State University of Science and Technology
Krasnoyarsk, Russian Federation
vladimirstanovov@yandex.ru

2nd Shakhnaz Akhmedova
Institute of Informatics and Telecommunication
Reshetnev Siberian State University of Science and Technology
Krasnoyarsk, Russian Federation
shahnaz@inbox.ru

3rd Eugene Semenkin
Institute of Informatics and Telecommunication
Reshetnev Siberian State University of Science and Technology
Krasnoyarsk, Russian Federation
eugenesemenkin@yandex.ru

***Abstract*—The single-objective numerical optimization is an important research field due to variety of real world applications. One of the most promising classes of numerical optimization algorithms is Differential Evolution. This paper proposes a new algorithm called RASP-SHADE to solve the CEC 2020 Bound Constrained Single Objective Optimization benchmark problems. The developed algorithm is based on the L-SHADE with Distance-based success history adaptation, incudes parameter adaptations of the jSO algorithm, and introduces several novelties. The ranking of population and archive according to fitness introduces the selective pressure, resulting in a new mutation strategy. A new archive update rule is applied with replacing only worst points and the parameters sampling scheme is changed. The experiments show that RASP-SHADE modifications result in significant improvements when compared to other state-of-the-art algorithms.**

***Index Terms*—optimization, differential evolution, selective pressure, CEC 2020**

I. Introduction

In recent decades the heuristic optimization methods have shown promising results for variety of real-world problems, with Differential Evolution (DE) being one of the most commonly used in case of numerical optimization. Since the first proposal of DE in [1], a variety of modifications have been proposed, which included novel mutation strategies and parameter adaptation mechanisms. Although the No Free Lunch theorem [2] for optimization states that no single optimization algorithm could be considered to be superior compared to others on all problems, it is still possible to develop efficient algorithms for specific classes of problems, which is improtat for practitioners. In most cases, the problem of constructing an efficient search algorithm is a problem of keeping the balance between exploration and exploitation.

In this paper the problem of tradeoff between exploration and exploitation in DE is addressed by several modifications. As a baseline, the L-SHADE algorithm originally presented in [3], further improved in jSO [4] and DISH [5]. As stated in [6], the L-SHADE class of algorithms represent one of the most efficient DE versions. First, the Success-History parameter adaptation is imrpoved by changing the sampling type for F and Cr parameters. Second, a novel mutation strategy, current-to-rank-arch/1 is proposed, which applies two separate fitness-based rankings for the population and the archive. Third, a new archive update rule is applied, according to which only inferior individuals are replaced in the archive set. The resulting algorithm is called Success Historty Adaptive Differential Evolution with Ranked Archive Selective Pressure and distance-based success-history adaptation (RASP-SHADE). The experimental results compare all proposed modifications and their combinations with the baseline algorithm, showing the effect of each of them.

The structure of the paper is organized as follows. In Section II a short description of L-SHADE, jSO and DISH is given. Section III presents the new modifications in detail, and the ides behind them. The experimental setup and results using

This work was supported by grant MK-1579.2020.9 given by President of Russian Federation for governmental support of the young researchers.

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

CEC 2020 benchmark problems is presented in Section IV. Finally, Section V contains results discussion, conclusions and directions of further research.

II. Related Work

A. Differential Evolution

The Differential Evolution is a widely used Evolutionary Algorithm (EA), which is capable of achieving highly competetive results for numerical optimization [7]. DE is a population based heuristic optimization method, which uses a set of solutions to perform the search. The main idea of DE is to use difference vectors between members of the population to generate new solutions. DE has three main parameters, population size NP, scaling factor F and crossover rate Cr. The algorithms starts by randomly initializing the population of vectors $x_{i,j}$, $i = 1...NP$, $j = 1...D$, inside the boundaries with uniform distribution, and then performs mutation, crossover and selection steps, where D is the problem dimension. In the original DE [1] the rand/1 mutation strategy was used, later more advanced strategies were proposed, including current-to-pbest/1, introduced in JADE algorithm [8] and also used in SHADE framework [9]. The current-to-pbest/1 strategy generates mutant vector using scaling factor $F \in [0, 1]$ as follows:

$$v_{i,j} = x_{i,j} = F(x_{pbest,j} - x_{i,j}) + F(x_{r1,j} - x_{r2,j}) \quad (1)$$

In (1) $x_{i,j}$ is the j-th coordinate of i-th individual, $pbest$ is the index of one of $pb * 100\%$ best individuals, different from index i, $r1$ and $r2$ are mutually different random indexes, also different from i and $pbest$. After generating the mutant vector, the crossover is performed to combine the genetic information. Most DE algorithms use binomial crossover, in which the trial vector u is created using parameter $Cr \in [0, 1]$ as follows:

$$u_{i,j} = \begin{cases} v_{i,j}, & \text{if } rand(0,1) < Cr \text{ or } j = jrand \\ x_{i,j}, & \text{otherwise} \end{cases} \quad (2)$$

In (2) the $jrand$ index is randomly chosen from $[1, D]$ and is required to make sure that at least one coordinate is taken from the mutant vector. Otherwise, it is possible there will be no difference between parent and child, and it will result in unnecessary fitness calculations. To keep solutions inside the boundaries, several bound constraint handling methods could be used. In most studies, the following bound constraint handling method (BCHM [10]) is applied: if the component of the trial vector violates the boundary, its parent is used to set the new value.

$$u_{i,j} = \begin{cases} \frac{xmin_j + x_{i,j}}{2}, & \text{if } u_{i,j} < xmin_j \\ \frac{xmax_j + x_{i,j}}{2}, & \text{if } u_{i,j} > xmax_j \end{cases}. \quad (3)$$

In (3) the $xmin_j$ and $xmax_j$ are the lower and upper boundaries of variable j. After calculating the fitness value $f(u)$ of the trial vector, the selection step is performed: if the trial vector is better or equal to the parent in terms of fitness, the i-th individual in the population is replaced.

$$x_{i,j} = \begin{cases} u_{i,j}, & \text{if } f(u_{i,j}) \leq f(x_{i,j}) \\ x_{i,j}, & \text{if } f(u_{i,j}) > f(x_{i,j}) \end{cases}. \quad (4)$$

Due to the greedy DE mutation, the fitness of the population either improves or stays the same.

B. L-SHADE algorithm

One of the main problems of DE is its high sensitivity to the parameter values. Following the ideas proposed in JADE, the L-SHADE algorithm, winner of CEC 2014 competition, has the following mechanisms for parameter adaptation. The historical memory cells $(M_{F,k}, M_{Cr,k})$ are maintained, each containing a couple of F and Cr parameter values. The memory size in L-SHADE is set to $H = 5$, h is the current memory index. For every mutation and crossover new parameter values are sampled as follows:

$$F = randc(M_{F,k}, 0.1), randn(M_{Cr,k}, 0.1), \quad (5)$$

where $randc$ is a Cauchy distributed random value, and $randn$ is normally distribured, k is chosen in range $[1, D]$ for every individual. The values in memory cells are used as location parameters of the distributions. If the generated F value is below 0, it is generated again, however if $F > 1$ it is set to 1, also if $Cr < 0$ then $Cr = 0$ and if $Cr > 1$ then $Cr = 1$.

The values of F and Cr, which delivered an improvement to the fitness of an individual, are stored in arrays S_F and S_{Cr}, as well as the fitness difference Δf, and used at the end of the generation to update the memory cells with weighted Lehmer mean:

$$mean_{wL} = \frac{\sum_{j=1}^{|S|} w_j S_j^2}{\sum_{j=1}^{|S|} w_j S_j}, \quad (6)$$

where $w_j = \frac{\Delta f_j}{\sum_{k=1}^{|S|} \Delta f_k}$, $\Delta f_j = |f(u_j) - f(x_j)|$ and S is either S_{Cr} or S_F. To avoid large changes in parameter values, the previous are used, so that the new memory cells values are defined as follows:

$$M_{F,k}^{g+1} = 0.5(M_{F,k}^{g} + mean_{wL}(F)), \quad (7)$$

$$M_{Cr,k}^{g+1} = 0.5(M_{Cr,k}^{g} + mean_{wL}(Cr)), \quad (8)$$

where g is the current generation number.

The JADE, SHADE, L-SHADE and their modifications use the external archive to store the solutions replaced during selection in an external archive A, usually of size NP. Initially the archive is empty and it is filled as the algorithm works. Every time a newly generated offspring replaces the parent, the later is copied into the archive. If the archive is full, the new individuals replace randomly selected ones. The archive is used during mutation, where the $r2$ index in current-to-pbest/1 strategy is randomly selected from the joined set of the population and the archive.

The L-SHADE algorithm controls the population size NP by linearly decreasing it (Linear Population Size Reduction,

LPSR). At the end of each generation the number of individuals to be deleted is determined, and the worst in terms are removed. The new population size is calculated as follows:

$$NP_{g+1} = round(\frac{NP_{min} - NP_{max}}{NFE_{max}} NFE + NP_{max}), \quad (9)$$

where $NP_{min} = 4$ is the minimal number of individuals in the population, NP_{max} is the initial population size, NFE is the current number of function evaluations, NFE_{max} is the maximal number of function evaluations. The idea of population size reduction is to allow wide search at the beginning to cover as much search space as possible, but later decrease the population size for better convergence to the best located optimum. The archive size NA is decreased in the same way as in (9). In the next section recent and most successful modifications of the L-SHADE algorithm are presented.

C. jSO and DISH algorithms

The jSO algorithm presented in [4] is the announced winner of the CEC 2017 competition on numerical optimization. The jSO introduces several new parameter control rules into the L-SHADE framework and uses the new mutation strategy, current-to-pbest-w/1, defined as follows:

$$v_{i,j} = x_{i,j} + F_w(x_{pb,j} - x_{i,j}) + F(x_{r1,j} - x_{r2,j}), \quad (10)$$

with F_w defined as follows:

$$F_w = \begin{cases} 0.7F, & \text{if } NFE < 0.2NFE_{max}, \\ 0.8F, & \text{if } 0.2NFE_{max} \leq NFE < 0.4NFE_{max}, \\ 1.2F, & otherwise. \end{cases} \quad (11)$$

The mutation strategy greediness is controlled by changing the pb value as follows: $pb = \frac{pb_{max} - pb_{min}}{NFE_{max}} NFE + pb_{min}$. The minimal value is set as $pb_{min} = pb_{max}/2$, with $pb_{max} = 0.25$ in jSO.

The crossover rate Cr is chaned so that large crossover rates are not allowed at the beginning of the search:

$$Cr = \begin{cases} max(Cr, 0.7), & \text{if } NFE < 0.25NFE_{max}, \\ max(Cr, 0.6), & \text{if } 0.25NFE_{max} \leq NFE \\ & \text{and } NFE < 0.5NFE_{max}. \end{cases} \quad (12)$$

The memory cells $M_{F,k}$ and $M_{Cr,k}$ are initialized with 0.3 and 0.8 in jSO in contrast to 0.5 in L-SHADE. In addition, one additional memory cell allways keeps the values of 0.9 for both F and Cr.

The DISH algorithm further improves the jSO by introducing the distance based weighting in the Lehmer mean calculation. The weights are determined not by the Δf, but by the Euclidean distance between the offspring u and parent x_i as follows:

$$w_n^E = \frac{\sqrt{\sum_{j=1}^{D}(u_{n,j} - x_{n,j})^2}}{\sum_{k=1}^{|S|}\sqrt{\sum_{j=1}^{D}(u_{k,j} - x_{k,j})^2}}. \quad (13)$$

This weight assignment method promotes exploration of the search space, so that if two points were far away from each other and there was an improvement, then the scaling parameter and crossover rate get larger weights. In the next section the proposed RASP-SHADE algorithm is described.

III. RASP-SHADE ALGORITHM

The RASP-SHADE uses all the presented features of the L-SHADE, jSO and DISH algorithms and follows the ideas used in L-SHADE-RSP [11], ranked 2nd best algorithm during the CEC 2018 competition. In [12] it was shown that the selective pressure is capable of improving the convergence capabilities of DE, especially for larger dimensions. The L-SHADE-RSP algorithm changes the probabilities of selecting individuals from the population for the r2 index in eq. (10), when an individual is chosen from the population, according to the fitness-based ranking. If the individual is chosen from the archive, no selective pressure is applied. The RASP-SHADE further develops this idea and uses the *current-to-ranked-archive/1* mutation strategy, defined as follows:

$$v_{i,j} = x_{i,j} + F_w(x_{r_1,j} - x_{i,j}) + F(x_{r_2,j} - x_{r_3,j}), \quad (14)$$

where F_w is defined in the same way as in eq. (11). Each of the indexes r_1, r_2 and r_3 is selected from the archive set with probability $p_A = \frac{|A|}{|A|+NP}$, otherwise selected from the population. The population and the archive receive independent rankings according to the fitness values. The ranks assignment could be performed in several ways, described in [12]. In this case, the linear ranking is used, so that $rank_i = i$, $i = 1...NP$, the largest rank assigned to the best individual (having smallest goal function value in case of minimization). The RASP-SHADE uses linear ranking for all three indexes r_1, r_2 and r_3. The probabilities of individuals being chosen are calculated as follows:

$$p_i^{sel} = \frac{rank_i}{\sum_{j=1}^{NP} rank_j}. \quad (15)$$

The probabilities for archive will be further reffered to as $p_{A,i}^{sel}$, and for population $p_{P,i}^{sel}$. It is possible to use non-linear ranking procedures, as reported in [12], for example exponential ranking, or use tournament selection. However, large selective pressure in DE usually leads to premature convergence.

The Cr parameter sampling scheme was changed, so that larger standard deviation $\sigma = 0.2$ is applied. So, in RASP-SHADE, the parameters are sampled as follows:

$$F = randc(M_{F,k}, 0.1), Cr = randn(M_{Cr,k}, 0.2). \quad (16)$$

The perofrmed experiments have shown that sampling Cr with parameters larger then 0.1 may improve convergence properties on several problems.

Another important modification used in RASP-SHADE is the archive update rule. Instead of replacing the first randomly chosen individual in the archive, the update rule in RASP-SHADE generates random indexes, untill an individual which is worse than the replaced parent, is found. If there is no such

individual, then a randomly selected individual is replaced. The search for the worst individual is performed randomly for $|A|$ iterations, i.e. depends on the current archive size.

The pseudocode of the RASP-SHADE algorithm is presented in Algorithm 1. The next section contains the experimental setup, parameter settings, results and discussion.

IV. Experimental Setup and Results

The experiments were performed in accordance with the CEC 2020 competition on single objective bound-constrained numerical optimization, presented in [13]. The benchmark contained 10 functions defined for $D = 5$, $D = 10$, $D = 15$ and $D = 20$. The computational resource is set to $5 \cdot 10^5$ function evaluations for $5D$, 10^6 for $10D$, $3 \cdot 10^6$ for $15D$ and 10^7 for $20D$. Functions 6 and 7 for $5D$ are excluded from the competition. Unlike previous competitions, in the CEC 2020 the amount of computational resource scales exponentially with the problem dimension, with the goal of determining the abilities of algorithms to correctly use the increased resource.

The was 30 independent algorithm runs performed, and the best function values were recorded after $D^{\frac{k}{5}-3}NFE_{max}$ function evaluations, k=0,...,15 for each test function. The population size was set to $NP = 30D^{1.5}$, maximum archive size $NA = 0.7NP$. The algorithm was implemented in C++ with GCC and run on PC with Ubuntu 19.04, Intel Core i7 8700k processor and 48GB RAM, results post-processing performed using Python 3.6.

The results of RASP-SHADE for all dimensions are presented in Tables I-IV.

TABLE I
Results of RASP-SHADE for $5D$

Func.	Best	Worst	Median	Mean	Std
1	0.0	0.0	0.0	0.0	0.0
2	0.023946	6.8356	0.364606	0.563579	1.175281
3	5.14823	5.7578	5.352665	5.337707	0.171504
4	0.004675	0.158992	0.110297	0.103801	0.038458
5	0.0	0.624099	0.0	0.0832132	0.212152
8	0.0	0.0	0.0	0.0	0.0
9	100.0	100.0	100.0	100.0	0.0
10	300.0	347.367	347.367	344.2092	11.815405

TABLE II
Results of RASP-SHADE for $10D$

Func.	Best	Worst	Median	Mean	Std
1	0.0	0.0	0.0	0.0	0.0
2	0.0624636	3.66478	0.312279	0.944419	1.316179
3	10.4359	11.7546	11.1883	11.140753	0.358079
4	0.177174	0.37063	0.276787	0.272218	0.057311
5	0.0	0.624429	0.416286	0.340375	0.173219
6	0.0211657	0.298681	0.196276	0.159378	0.093905
7	8.724e-07	0.004154	0.000771	0.000894	0.000958
8	100.0	100.0	100.0	100.0	0.0
9	100.0	100.0	100.0	100.0	0.0
10	397.743	398.009	397.743	397.760733	0.066352

The RASP-SHADE algorithm was able to find the exact solution for the first function (unimodal bent cigar) in all

Algorithm 1 RASP-SHADE

Set NP_{max}, $NFE = 0$, $NP = NP_{max}$ D, NFE_{max},
$H = 5$, $A = \emptyset$, $M_{F,r} = 0.5$, $M_{Cr,r} = 0.8$, $r = 1...H$
$M_{F,H+1} = 0.9$, $M_{Cr,H+1} = 0.9$, $NA = 0.7NP$
$g = 0$
Initialize population $P_0 = (x_{1,j}, ..., x_{NP,j})$ randomly
while $NFE < NFE_{max}$ **do**
  $S_F = \emptyset$, $S_{Cr} = \emptyset$
  Rank population according to fitness, set p_P^{sel}
  Rank archive according to fitness, set p_A^{sel}
  for $i = 1$ to NP **do**
    Current memory index $r = randint[1, H + 1]$
    $Cr = randn(M_{Cr,r}, 0.2)$
    $Cr = min(1, max(0, Cr))$
    repeat
      $F = randc(M_{F,r}, 0.1)$
    until $F \geq 0$
    $F = min(1, F)$
    Get F_w from eq. (11)
    Limit Cr according to eq. (12)
    repeat
      for k=1 to 3 **do**
        if $rand[0, 1] < \frac{|A|}{|A|+NP}$ **then**
          r_k from archive with p_A^{sel}
        else
          r_k from population with p_P^{sel}
        end if
      end for
    until $i \neq r_1 \neq r_2 \neq r_3$
    for j=1 to D **do**
      $v_{i,j} = x_{i,j} + F_w(x_{r_1,j} - x_{i,j}) + F(x_{r_2,j} - x_{r_3,j})$
    end for
    Get u_i from eq. (2) with Cr, calculate $f(u_i)$
    if $f(u_i) < f(x_i)$ **then**
      repeat
        $r_A = randInt[1, |S|]$
      until $f(r_A) > f(x_i)$ or NP attempts made
      $x_i \rightarrow A$, $x_i = u_i$, $F \rightarrow S_F$, $Cr \rightarrow S_{Cr}$
    end if
  end for
  Get NP_{g+1} from eq. (9)
  Recalculate maximum archive size NA_{g+1}
  if $|A| > NA_{g+1}$ **then**
    Remove random individuals from the archive
  end if
  if $NP_g > NP_{g+1}$ **then**
    Remove worst individuals from the population
  end if
  if $S_F \neq \emptyset$ and $S_{Cr} \neq \emptyset$ **then**
    Update $M_{F,k}$ and $M_{Cr,k}$ with eq. (13)
  end if
  $k = k + 1$, $g = g + 1$
  if $k > H$ **then**
    $k = 1$
  end if
end while
Return best solution x_{best}

TABLE III
RESULTS OF RASP-SHADE FOR $15D$

Func.	Best	Worst	Median	Mean	Std
1	0.0	0.0	0.0	0.0	0.0
2	0.083274	4.71995	0.271543	1.023448	1.334323
3	15.567	16.3923	15.72265	15.808616	0.228542
4	0.240077	0.46057	0.37005	0.363859	0.045738
5	0.468353	3.45323	1.38525	1.325446	0.837661
6	0.237946	1.14647	0.678188	0.616672	0.208141
7	0.499952	0.91628	0.708143	0.728940	0.124301
8	100.0	100.0	100.0	100.0	0.0
9	300.0	386.085	300.0	333.180666	40.757435
10	400.0	400.0	400.0	400.0	0.0

TABLE IV
RESULTS OF RASP-SHADE FOR $20D$

Func.	Best	Worst	Median	Mean	Std
1	0.0	0.0	0.0	0.0	0.0
2	0.031228	0.218599	0.124914	0.138446	0.045385
3	20.3872	21.2422	20.4979	20.530463	0.188792
4	0.348645	0.561976	0.459255	0.453013	0.041777
5	0.312256	5.39114	1.30721	1.413648	1.249984
6	0.050391	0.310306	0.162172	0.170902	0.057422
7	0.171816	1.1439	0.831667	0.726503	0.240382
8	100.0	100.0	100.0	100.0	0.0
9	300.0	383.742	360.523	341.7386	35.123196
10	413.657	413.657	413.657	413.656999	1.1368e-13

dimensions, as well as for hybrid function 8 in $5D$ case. Also, exact solutions were sometimes found for hybrid function 5 in $5D$ and $10D$, and at least one relatively good solution was found for hybrid function problem 7 in $10D$. For composition functions 8-10 in most cases the RASP-SHADE algorithm was able to find suboptimal solutions.

To compare the results of RASP-SHADE with other approaches, several experiments have been performed. First, as long as RASP-SHADE is a modification of jSO, DISH and LSHADE-RSP algorithms, these methods were tested. The population sizes for all these approaches were set in the same manner, i.e. $NP = 30D^{1.5}$, all other parameters were set as described in the original papers, exept for the pb_{max} in LSHADE-RSP set to 0.25 instead of 0.17. In addition to this, all modifications introduced in the RASP-SHADE were tested. The comparison between methods was performed using two-tailed Mann-Whitney rank sum statistical test with significance level $p = 0.01$, normal approximation and tie correction. The statistical test was performed for every function independently based on 30 resulting values at the end of each run, the Z and p value were reported. Table V shows the comparison of jSO, DISH, LSHADE-RSP and three versions of RASP-SHADE: with random archive update and $\sigma = 0.1$ for Cr sampling (RA, 0.1), with new archive update searching for worst individual and $\sigma = 0.1$ (WA, 0.1), and with new archive update rule and $\sigma = 0.2$ (WA, 0.2). All algorithms are compared to jSO, which was taken as a baseline.

In Table V the total number of wins (+), non-significant differences (=) and losses (-) summed over all functions ac-

TABLE V
STATISTICAL COMPARISON OF RASP-SHADE WITH OTHER APPROACHES, MANN-WHITNEY TEST, JSO AS BASELINE

D	$5D$	$10D$	$15D$	$20D$
DISH	0+/8=/0-	0+/10=/0-	1+/9=/0-	1+/9=/0-
LSHADE-RSP	0+/8=/0-	0+/10=/0-	0+/10=/0-	1+/9=/0-
RASP-SHADE$_{RA,0.1}$	0+/7=/1-	2+/7=/1-	1+/6=/3-	1+/9=/0-
RASP-SHADE$_{WA,0.1}$	0+/7=/1-	3+/7=/0	1+/9=/0-	3+/7=/0-
RASP-SHADE$_{WA,0.2}$	0+/8=/0	4+/6=/0-	1+/9=/0-	3+/7=/0-

cording to Mann-Whitney test are reported for all algorithms. On CEC 2020 problems applying the Euclidean distance based parameter adaptation, introduced in DISH algorithm only gives significant improvement for one problem in $15D$ and $20D$, namely function 5. The LSHADE-RSP algorithm, which does not use the Euclidean distance weighting, is still capable of showing better performance then jSO for one $20D$ function, namely F9. The usage of other mutation strategy in the RASP-SHADE algorithm changes its behaviour, resulting in several performance improvements and deteriorations. Adding the new archive update rule and larger σ for Cr sampling improves the performance, resulting in up to 4 improvements for $10D$ and 3 improvements for $20D$, in particular for functions 5, 6 and 9.

In addition to the Mann-Whitney test, the ranking procedure from Friedman statistical test was used to compare the algorithms. All 6 algorithms' results were ranked on every function and every dimension independently, and the sum of all ranks was averaged aand summed over all functions, with smaller ranks assigned to better goal function values. Table VI shows the final ranks of all used algorithms.

TABLE VI
COMPARISON OF RASP-SHADE WITH OTHER APPROACHES, FRIEDMAN RANKING

D	$5D$	$10D$	$15D$	$20D$
jSO	**14.1960**	16.2745	15.0980	17.1960
DISH	14.7647	16.5490	14.3921	14.9509
LSHADE-RSP	14.2254	15.6764	14.7941	16.3529
RASP-SHADE$_{RA,0.1}$	15.3137	15.1568	16.2843	15.7941
RASP-SHADE$_{WA,0.1}$	15.3529	12.4803	14.3921	12.0098
RASP-SHADE$_{WA,0.2}$	14.3823	**12.0980**	**13.2745**	**11.9313**

From Table VI it could be seen that for $5D$ the jSO is better than other approaches, but the difference is small, while the last modification of RASP-SHADE achieves the best results for other dimensions. Also, it could be mentioned that the modified archive update rule has the largest effect on the performance.

Table VII shows the complexity analysis of RASP-SHADE algorithm, performed according to the CEC 2020 competition rules.

The most computationally intensive new modification in the RASP-SHADE is the fitness-based ranking and probabilities assignment for the *current-to-ranked-archive/1* mutation strat-

egy. The contribution of other parts, including the archive update rule, is minor.

In Figures 1 and 2 the convergence graphs of jSO, DISH, LSHADE-RSP and RASP-SHADE$_{WA,0.2}$ are presented for 10-dimensional and 20-dimensional functions.

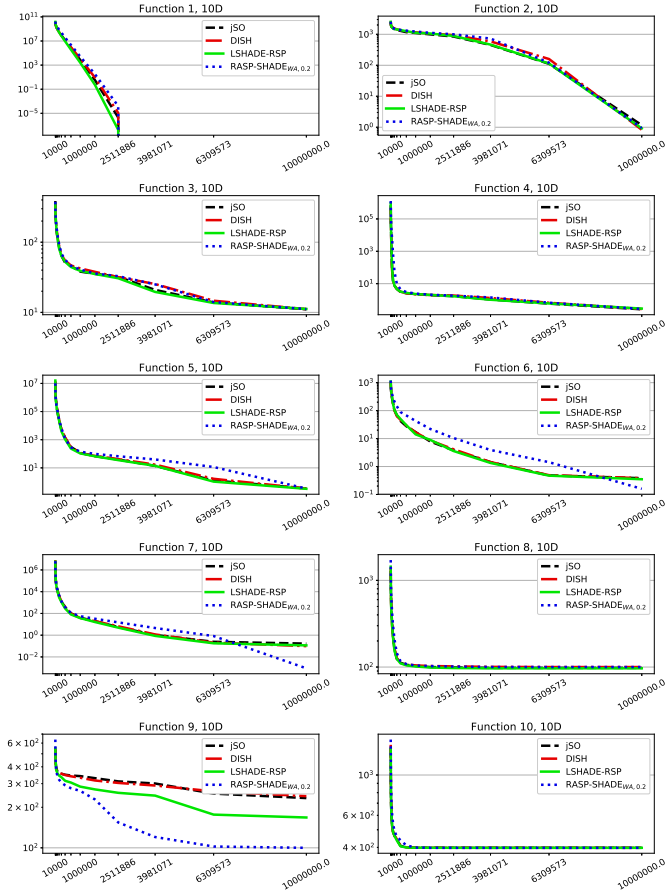

Fig. 1. Convergence graphs for all functions, $10D$.

TABLE VII
COMPUTATIONAL COMPLEXITY OF RASP-SHADE

D	$T0$	$T1$	$T2$	$(T2-T1)/T0$
$D=5$	0.303	0.024	190.8	0.3326
$D=10$	0.303	0.035	262.2	0.5683
$D=15$	0.303	0.059	340.2	0.8267
$D=20$	0.303	0.090	436.4	1.1432

From Figures 1 and 2 it could be seen that RASP-SHADE demonstrates different convergence properties, especially for functions 5, 6, 7 and 9, which are complex hybrid and composition functions. The RASP-SHADE converges slower, however it maintains its explorative properties much longer, allowing to achieve better performance in several scenarios. The main reason of the improved explorative properties is the *current-to-ranked-archive/1* mutation strategy, and the convergence speed is improved by introducing the selective pressure and new archive update rule.

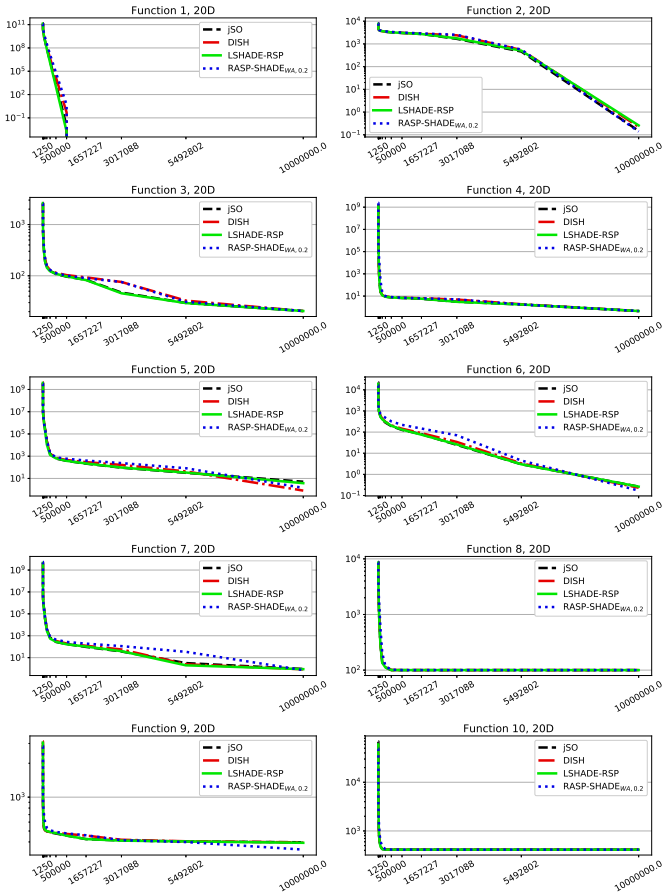

Fig. 2. Convergence graphs for all functions, $20D$.

V. CONCLUSIONS

The RASP-SHADE algorithm introduced in this paper has several significant differences, compared to the L-SHADE class of algorithms. First is the novel mutation strategy, which uses both the population and the archive for all three vectors in the equation, improving the exploration properties. The selective pressure, on the other hand, is aimed at exploiting the known information, while also taking into consideration the information contained in the archive. Second, the new archive update rule is focused on saving the information in the archive by replacing only inferior solutions. With increased crossover rate scalinig parameter, the RASP-SHADE outperforms several state-of-the-art algorithms, which were considered one of the best on previous CEC competitions. The directions of further work may include the search for new mutation strategies and selective pressure mechanisms, understanding their behaviour, as well as new archive set handling techniques.

REFERENCES

[1] R. Storn and K. Price, Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, Journal of Global Optimization, Vol. 11, N. 4, pp. 341-359, 1997, doi: 10.1023/A:1008202821328

[2] Wolpert, D. H., and Macready, W. G. (1997). No free lunch theorems for optimization. IEEE transactions on evolutionary computation, 1(1), 67-82.

[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] 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.

[5] Viktorin, A., Senkerik, R., Pluhacek, M., Kadavy, T., and Zamuda, A. Distance based parameter adaptation for Success-History based Differential Evolution. Swarm and Evolutionary Computation. Volume 50, 2019, https://doi.org/10.1016/j.swevo.2018.10.013

[6] Al-Dabbagh, R. D., Neri, F., Idris, N., and Baba, M. S. Algorithmic design issues in adaptive differential evolution schemes: Review and taxonomy. In Swarm and Evolutionary Computation 43, pp. 284–311 (2018)

[7] 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.

[8] 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.

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

[10] R. Biedrzycki, J. Arabas, D. Jagodziński, ”Bound constraints handling in Differential Evolution: An experimental study”, Swarm and Evolutionary Computation, Vol. No. 50, 2019.

[11] V. Stanovov, S. Akhmedova and E. Semenkin, ”LSHADE Algorithm with Rank-Based Selective Pressure Strategy for Solving CEC 2017 Benchmark Problems,” 2018 IEEE Congress on Evolutionary Computation (CEC), Rio de Janeiro, 2018, pp. 1-8. doi: 10.1109/CEC.2018.8477977

[12] V. Stanovov, S. Akhmedova and E. Semenkin, Selective Pressure Strategy in differential evolution: Exploitation improvement in solving global optimization problems, Swarm and Evolutionary Computation, Volume 50, 2019, ISSN 2210-6502, doi: 10.1016/j.swevo.2018.10.014.

[13] C. T. Yue, K. V. Price, P. N. Suganthan, J. J. Liang, M. Z. Ali, B. Y. Qu, N. H. Awad and P. Biswas, Problem Definitions and Evaluation Criteria for the CEC 2020 Special Session and Competition on Single Objective Bound Constrained Numerical Optimization, Technical Report 201911, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore, November 2019.