NL-SHADE-LBC algorithm with linear parameter adaptation bias change for CEC 2022 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*—In this paper the adaptive differential evolution algorithm is presented, which includes a set of concepts, such as linear bias change in parameter adaptation, repetitive generation of points for bound constraint handling, as well as non-linear population size reduction and selective pressure. The proposed algorithm is used to solve the problems of the CEC 2022 Bound Constrained Single Objective Numerical Optimization benchmark problems. The computational experiments and analysis of the results demonstrate that the NL-SHADE-LBC algorithm presented in this study is able to demonstrate high efficiency in solving complex optimization problems compared to the winners of the previous years' competitions.**

***Index Terms*—optimization, differential evolution, parameter adaptation, CEC 2022**

I. Introduction

Most of the problems presented in different scientific and engineering areas can be formulated as optimization problems [1]. There are two big groups of methods developed to solve them, namely, classical approaches, which are based on objective function's features, and the heuristic approaches, which only need the objective function values. Classical approaches are not generally efficient in solving complex optimization problems due to their dependency on preconditions such as continuity, convexity and differentiability of objective function, that are not usually met especially in case of real-world optimization problems.

Heuristics do not demonstrate most of the drawbacks of classical approaches and, thus, they are frequently used in real-case scenarios. Heuristic approaches include a large class of evolutionary and biology-inspired algorithms and the Differential Evolution (DE) is one of the most efficient approaches for solving numerical optimization problems [2]. The DE algorithm has mutation, crossover and selection operators [3] similarly to other evolutionary algorithms, but it is much easier to implement, which makes it so popular among researchers. However, it was shown multiple times that additional parameter control and adaptation techniques can significantly improve DE's performance.

In this study a new version of the L-SHADE approach [4], which is in its turn a well-known and popular modification of the differential evolution algorithm, is proposed. A new algorithm was called NL-SHADE-LBC (Non-Linear population size reduction Success-History Adaptive Differential Evolution with Linear Bias Change) and it was used for the IEEE CEC 2022 competition on real parameter single objective bound constrained numerical optimization [5]. The proposed algorithm is an improved variant of the NL-SHADE-RSP approach [6], which was among the winners of the IEEE CEC 2021 competition on numerical optimization [7]. It combines selective pressure, biased parameter adaptation with linear bias change (which is one of the main features), current-to-pbest strategy, resampling of solutions as bound constraint handling techniques, as well as the non-linear population size reduction (NLPSR). The NL-SHADE-LBC approach was compared to the top 4 algorithms from CEC 2021 competition, namely to NL-SHADE-RSP [6], MLS-LSHADE [8], MadDE [9] and APGSK-IMODE [10]. The comparison was conducted by using statistical tests such as the Friedman ranking test and the Mann-Whitney rank sum test.

The structure of this paper is organized as follows. In Section II a short description of the differential evolution algorithm and its modification L-SHADE is given. Section III presents the new approach in detail, and the ideas behind

This work was supported by the Ministry of Science and Higher Education of the Russian Federation within limits of state contract № FEFE-2020-0013.

them. The experimental setup and results obtained for the CEC 2022 benchmark problems are presented in Section IV. Finally, Section V contains results discussion and conclusions.

II. Related Work

A. Differential Evolution

The Differential Evolution algorithm is an evolutionary method, which was introduced in [3] by Storn and Price for solving real-valued optimization problems. It works with a population of potential solutions (individuals) randomly initialized in the search space. To be more specific, let us assume that there are NP individuals in the population, thus, NP is the population size, which is the first parameter of the DE approach. Each i-th individual is represented as vector $x_i = (x_{i,1}, x_{i,2}, \dots, x_{i,D})$ in the search space, where D is the problem dimensionality. So firstly $x_{i,j}$, $i = 1...NP$, $j = 1...D$, is generated within the range $[xmin_j, xmax_j]$ by using the uniform distribution.

The DE approach similarly to other evolutionary algorithms has three basic operators, namely, mutation, crossover and selection. The idea behind the DE algorithm is to use the difference vectors between individuals currently existing in the population. These difference vectors are used during the mutation step to generate new vectors called mutants with aim to move population closer to the optimal solution. The second parameter of the DE approach, namely the scaling factor F, is used for applied mutation strategy. Although the rand/1 mutation strategy was proposed in the original work, nowadays approaches based on differential evolution mostly use the current-to-pbest/1 strategy, which was introduced for the JADE algorithm [11]. The current-to-pbest strategy works as follows:

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

Here $pbest$ is an index of one of the $p*100\%$ best individuals in the current population, $r1$, $r2$ are randomly chosen indexes from the population and scaling factor is F usually in the range $[0, 1]$. It should be noted that indexes $pbest$, $r1$ and $r2$ are generated different from i and each other.

On the next step the crossover operator is applied to generate new individuals, called trial vectors, by combining the information contained in the current population and mutant vectors. The crossover rate Cr is introduced for this operator and it is the last parameter of the DE approach.

The binomial crossover is commonly used for the differential evolution algorithm. During that crossover the trial vector u_i receives randomly chosen components from the mutant vector with probability $Cr \in [0, 1]$ as follows:

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

In the last formula $jrand$ is a randomly chosen index from $[1, D]$, which is required to make sure that the trial vector is different from the target vector to avoid unnecessary fitness calculations.

All the generated trial vectors u have to be within a search space, therefore, the bound constraint handling method (BCHM) is applied to them. There are various well-known BCHM [12] and in this study the resampling of solutions in combination with the midpoint target approach was used. The midpoint target approach can be described as follows:

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

The final step of the differential evolution algorithm is selection. It is performed after calculating the objective function values $f(u)$. To be more specific, if the trial vector u_i outperforms or is equal to the vector x_i from the current population in terms of the objective function, then the target vector x_i in the population is replaced by the trial vector u_i:

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

So by the end of each iteration population is updated and all the mentioned steps are performed until some stopping criterion is met.

B. L-SHADE algorithm

The DE approach is well-known for its simplicity due to the fact that unlike most of other evolutionary algorithms it has just a few number of parameters such as the population size NP, the scaling factor F and the crossover rate Cr. However, it was shown that the DE algorithm is very sensitive to its parameters and, thus, they should be carefully chosen for each problem, otherwise obtained results could be far from the good ones. Therefore, the development of the improved DE variants mainly focused on the parameter adaptation issue. Significant progress in the field of adaptation strategies was achieved when the JADE algorithm [11] was first introduced. It was followed by the development of the SHADE approach [13] and its modification L-SHADE [4], where the latter is still considered as one of the most efficient evolutionary algorithms. So in this subsection the main features of L-SHADE algorithm, which is one of the most popular DE-based optimizers [14] are described.

The L-SHADE algorithm uses a set of $H = 5$ historical memory cells containing values $(M_{F,k}, M_{Cr,k})$ to generate new parameter values for mutation and crossover steps on each iteration. These parameter values are sampled using randomly chosen memory index $k \in [1, H]$ as follows:

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

Here $randc(m, s)$ is the random number generated by Cauchy distribution, and $randn(m, s)$ is the normally distributed random value with position parameter m and scale parameter s.

The Cr value should be in the range $[0, 1]$, however, if that is not the case then Cr is set to 0 if the obtained value is less than 0 or 1 if it is greater than 1. The F value should also be in the range $[0, 1]$, and it is set to 1 if $F > 1$, but it is generated again if $F < 0$. The latter can be explained by the fact that there will not be an actual mutation in case if $F = 0$. Then the values of F and Cr, which delivered an improvement to an individual, are saved into arrays S_F and S_{Cr} together with the fitness difference Δf. The memory cell with index h incrementing from 1 to H every generation is updated in the following way:

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

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.

The previous parameter values are used to set the new ones with update parameter c as follows:

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

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

where g is the current generation number. In L-SHADE algorithm the update parameter is set to $c = 0.5$.

The L-SHADE algorithm uses the Linear Population Size Reduction (LPSR), so the population size NP is recalculated at the end of the iteration, and the worst individuals are removed from the population. Thus, the number of individuals decreases linearly depending on the available computational resource in the following way:

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

where $NP_{min} = 4$ and NP_{max} are the minimal and initial population sizes, NFE and NFE_{max} are the current and maximal number of function evaluations.

Additionally, the L-SHADE algorithm uses an external archive A, which contains parent solutions rejected during the selection step, and it is filled until its size reaches the predefined value NA. If the number of individuals in the archive reaches NA, then new solutions replace randomly selected ones in the archive. Also the current-to-pbest mutation strategy was changed to use the individuals from the external archive so that the $r2$ index is taken either from the population or the archive with a given probability of 0.5. The archive size decreases together with the population size, following the same LPSR rule.

C. L-SHADE modifications and alternative approaches

The main ideas of the L-SHADE algorithm were used as basis for developing similar but more efficient approaches. For example, the L-SHADE-RSP algorithm [15] was ranked as the second best algorithm in the CEC 2018 competition on numerical optimization [16] while it was the best non-hybrid DE-based approach among winners. The main difference between the L-SHADE-RSP approach and the original L-SHADE algorithm is that in L-SHADE-RSP the rank-based selective pressure was implemented. The latter caused changes in the mutation strategy, and the new one was called current-to-pbest/r. So the probabilities pr of indexes $r1$ and $r2$ were changed when chosen from the population in the following way:

$$pr_i = \frac{R_i}{\sum_{j=1}^{NP} R_j}. \quad (10)$$

Here ranks $R_i = e^{-i/NP}$ were set as indexes of individuals in an array sorted by fitness values, with largest ranks assigned to best individuals, $i = 1...NP$.

For the CEC 2021 competition on numerical optimization the following four algorithms were considered as the best ones: NL-SHADE-RSP, MLS-LSHADE, MadDE and APGSK-IMODE. It should be noted that all of the mentioned algorithms are DE-based methods.

The NL-SHADE-RSP approach [6] is a further development of the L-SHADE-RSP algorithm. It combines several novel parameter control techniques, such as Non-Linear Population Size Reduction (NLPSR), as well as Linear Crossover Rate Increase (LCRI). Moreover, a new strategy adaptation technique was used to choose between two mutation strategies during the search process, namely, with and without external archive.

A Multi-start Local Search Algorithm with L-SHADE or MLS-LSHADE [8] is an iterative algorithm, where each iteration is divided into two phases. In the first phase the L-SHADE algorithm is adopted to construct a good starting point for the later phase. After that a local search procedure is invoked in the second phase aiming to refine the result of the former phase and to locate the optimal solution.

The MadDE approach [9] is a modification of the DE algorithm that uses a multiple adaptation strategy for simultaneously adapting the control parameters and search mechanisms. It employs multiple strategies for mutation and crossover to generate trial vectors. The idea is that multiple search strategies are likely to give an overall consistent performance across a variety of objective functions with different properties. The MadDE algorithm improves on existing mutation strategies for carrying out the search and selects them in a probabilistic manner. The probability of selecting a mutation strategy depends on its historical rate of generating successful solutions. It also uses a probabilistic crossover technique. Additionally, it adapts the DE control parameters (NP, F and Cr) following the LSHADE algorithm.

The APGSK-IMODE algorithm [10] is the hybridization between the gaining sharing knowledge based algorithm with adaptive parameters (APGSK) [17] and IMODE algorithm [18]. In the APGSK-IMODE approach both algorithms (APGSK and IMODE) guide to different sub-populations to achieve the optimal or near-optimal solutions. Algorithm's framework emphasizes the best performing algorithm of APGSK and IMODE by updating the probability of selecting each of them based on the solutions quality during the evolutionary process.

Based on the features introduced in the mentioned algorithms, the NL-SHADE-LBC approach is proposed and described in details in the next section.

III. NL-SHADE-LBC ALGORITHM

The NL-SHADE-LBC algorithm is a further development of L-SHADE-RSP and NL-SHADE-RSP approaches, presented in [15] and [6] and uses the idea of applying selective pressure, studied in [19] and biased parameter adaptation, presented in [20]. NL-SHADE-LBC uses the current-to-pbest strategy same as many other SHADE-based approaches, but it also applies the rank-based selective pressure to $r2$ index if it is chosen from the population. If $r2$ is chosen from the archive, no selective pressure is applied. In NL-SHADE-LBC the ranks are assigned with a larger coefficient $R_i = e^{-4i/NP}$, resulting in larger selective pressure.

One of the features of NL-SHADE-LBC inherited from NL-SHADE-RSP is the non-linear population size reduction (NLPSR), originally proposed in AGSK algorithm. The NLPSR results in smaller population sizes compared to linear population size reduction given the same number of initial and final population sizes [6].

The NL-SHADE-LBC uses fixed archive usage probability, i.e. if the archive is not empty, then the probability to choose an individual from the archive for $r2$ index is set to 0.5. Only the binomial crossover is used in NL-SHADE-LBC. The pb parameter in the mutation is gradually increased (inspired by jSO [21]), i.e. the initial value of pb_{min} is set to 0.2, and the number of individuals to choose $pbest$ from is linearly increased as follows:

$$pbest = max(2, NP(0.2 + 0.1\frac{NFE}{NFE_{max}})). \tag{11}$$

Note that the minimal value of pb is 2 individuals. Increasing pb allows compensating the reduced population size during the later stages of the search.

Another feature of the NL-SHADE-LBC approach is the usage of biased parameter adaptation using generalized Lehmer mean, originally studied in [22]. The Lehmer mean applied in the JADE algorithm for scaling factor F memory update only, was further applied in SHADE for both F and Cr values. The generalized form of weighted Lehmer mean [23] is as follows:

$$mean_{w,p,m}(S) = \frac{\sum_{j=1}^{|S|} w_j S_j^p}{\sum_{j=1}^{|S|} w_j S_j^{p-m}} \tag{12}$$

where w_j are weights, $j = 1, ..., |S|$, and p and m are the parameter values. The equation defines a set of means, and changing the parameters allows getting different ways of mean calculation. For example, $p = 0$ and $m = 1$ results in harmonic mean, $p = 0.5$ and $m = 1$ - geometric mean, $p = 1$ and $m = 1$ - arithmetic mean, $p = 2$ and $m = 1$ - contraharmonic mean (used by SHADE). In most cases changing the p parameter is sufficient to get a different mean, however changing m is also possible, resulting in biased weighted means. The difference between means with $m = 1$ and $m = 1.5$ is shown in Figure 1, a uniform grid of values from 0 to 1 with equal weights is used as an input.

The NL-SHADE-LBC algorithm applies the linear bias change for both scaling factor F and crossover rate Cr. For both cases $m = 1.5$, and the initial value for F memory cell update is set to $p_{ini,F} = 3.5$, and the initial value for Cr memory cell update is set to $p_{ini,Cr} = 1.0$. The final values are set to $p_{fin,F} = p_{fin,Cr} = 1.5$. These values for both F and Cr are updated as follows:

$$p_g = round(\frac{(NFE_{max} - NFE)(p_{ini} - p_{fin})}{NFE_{max}}) + p_{fin}, \tag{13}$$

where g is the current generation number, at which the update is performed. According to this equation, the p parameter for F starts at 3.5 and decreases to 1.5, and for Cr it starts at 1.0 and increases to 1.5.

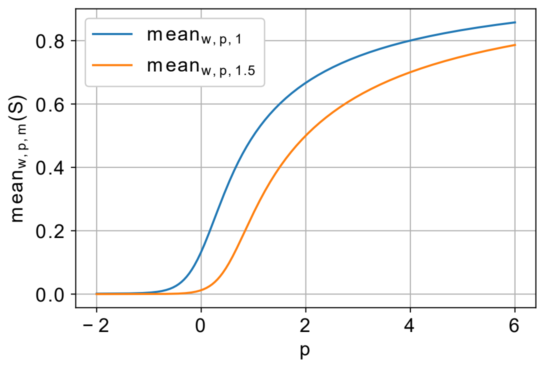

Fig. 1. Generalized Lehmer mean depending on parameter p with $m = 1$ and $m = 1.5$

In addition to the described features, the NL-SHADE-LBC uses modified bound constraint handling technique. Whenever a mutant vector is generated outside the boundaries, it is generated again, with a newly sampled F parameter, as well as $pbest$, $r1$ and $r2$ indexes. The resampling procedure is repeated up to 100 times, and if the new point is still outside the boundaries, then it is handled with midpoint target method.

Following the findings in [24], the archive update strategy is changed. In particular, the following scheme is used: if the archive is full, for the newly placed point a point from the archive is randomly selected, and if the fitness of the new point is better than of the selected one, the replacement occurs. Otherwise, another random point is selected, and the process continues up to the number of steps, equal to the archive size. If no point is found, the new one replaces a random one regardless of the fitness values. Note that this also requires the fitness values of the points in the archive to be stored.

The pseudocode of the NL-SHADE-LBC algorithm is presented in Algorithm 1. The parameters of NL-SHADE-LBC, such as population size, were tuned in several computational experiments. The next section contains the experimental setup, parameter settings, results and discussion.

IV. EXPERIMENTAL SETUP AND RESULTS

The experiments with the NL-SHADE-LBC approach were performed according to the rules of the IEEE Congress on Evolutionary Computation 2022 Competition on Single Objective Bound Constrained Numerical Optimization [5]. The CEC

Algorithm 1 NL-SHADE-LBC

```
1: Set $NP_{max} = 23D$, $NP = NP_{max}\ D$, $NFE_{max}$,
2: $H = 20D$, $A = \emptyset$, $M_{F,r} = 0.5$, $M_{Cr,r} = 0.9$, $k = 1$
3: $NA = 1.0NP$, $n_A = 0.5$, $g = 0$
4: Initialize population $P_0 = (x_{1,j}, ..., x_{NP,j})$ randomly
5: while $NFE < NFE_{max}$ do
6:    $S_F = \emptyset$, $S_{Cr} = \emptyset$, $n_A = 0$
7:    Rank population according to fitness $f(x_i)$
8:    for $i = 1$ to $NP$ do
9:       Current memory index $r = randInt[1, H + 1]$
10:      Crossover rate $Cr_i = randn(M_{Cr,r}, 0.1)$
11:      $Cr_i = min(1, max(0, Cr_i))$
12:   end for
13:   Sort $Cr_i$ according to fitness $f(x_i)$
14:   for $i = 1$ to $NP$ do
15:      $r = 0$
16:      repeat
17:         repeat
18:            $F_i = randc(M_{F,r}, 0.1)$
19:         until $F_i \geq 0$
20:         $F_i = min(1, F_i)$
21:         repeat
22:            $pbest = randInt(1, NP * p)$
23:            $r1 = randInt(1, NP)$
24:            if $rand[0, 1] < p_A$ then
25:               $r2 = randInt(1, NP)$
26:            else
27:               $r2 = randInt(1, NA)$
28:            end if
29:         until $i \neq pbest \neq r1 \neq r2$
30:         for j=1 to D do
31:            $v_{i,j} = x_{i,j} + F(x_{pbest,j} - x_{i,j}) + F(x_{r1,j} - x_{r2,j})$
32:         end for
33:         Binomial crossover with $Cr_b$, $r = r + 1$
34:      until $u_i < xmax$ and $u_i > xmin$ or $r \geq 100$
35:      Apply midpoint target BCHM to $u_i$
36:      Calculate $f(u_i)$, $ca = 0$
37:      if $f(u_i) < f(x_i)$ then
38:         repeat
39:            $ra = randInt[1, NA]$, $ca = ca + 1$
40:         until $f(A_{ra}) < f(x_i)$ or $ca = NA$
41:         $x_i \to A_{ra}$, $x_i = u_i$
42:         $F \to S_F$, $Cr \to S_{Cr}$, $\Delta f_i = f(x_i) - f(u_i)$
43:      end if
44:   end for
45:   Get $NP_{g+1}$ and $NA_{g+1}$ with NLPSR
46:   if $|A| > NA_{g+1}$ then
47:      Remove random individuals from the archive
48:   end if
49:   if $NP_g > NP_{g+1}$ then
50:      Remove worst individuals from the population
51:   end if
52:   Update $M_{F,k}$, $M_{Cr,k}$
53:   $k = mod(k, H) + 1$, $g = g + 1$
54: end while
55: Return best solution $x_{best}$
```

2022 benchmark contains 12 test functions for $D = 10$ and $D = 20$, with the maximum number of function evaluations set to $NFE_{max} = 2 \cdot 10^5$ and $NFE_{max} = 10^6$ respectively. Unlike the previous year's competition, there is only one set of functions, which are biased, shifted and rotated. During the 30 independent runs on every function, the best achieved values were saved after $D^{\frac{k}{5}-3}NFE_{max}$ evaluations of the goal function, $k = 0, ..., 15$. The goal of the CEC 2022 competition is to determine not only the best algorithm in terms of best achieved values at the end of the search, but also determine the fastest algorithms, i.e. the ones achieving the solution earlier. The NL-SHADE-LBC algorithm was implemented in C++, compiled with GCC 9.3.0 under Ubuntu Linux 20.04, with AMD Ryzen 5800X processor. The results post-processing, ranking and statistical tests were performed with Python 3.8.

To compare NL-SHADE-LBC with alternative approaches, the algorithms from CEC 2021 competition, for which the source code was available, were chosen. In particular, the NL-SHADE-RSP [6], the APGSK-IMODE [10], the MadDE [9] and the MLS-SHADE [8] were tested. In addition, the L-SHADE-RSP approach [15], which took the second place in CEC 2018 competition, was considered. The parameters of these algorithms were not changed, except for the benchmark settings i.e. number of functions, runs, computational resource. The MadDE approach used the SUBHO strategy to tune the hyperparameters of the algorithm, and for comparison the hyperparameters were not tuned for CEC 2022 benchmark, i.e. they were set as presented for CEC 2021 benchmark. To compare the algorithms the Mann-Whitney rank sum test with tie-breaking and normal approximation was applied, with significance level $p = 0.01$. The comparison between the listed methods for $10D$ and $20D$ functions is presented in Tables I.

TABLE I
MANN-WHITNEY TESTS OF NL-SHADE-LBC AGAINST OTHER APPROACHES, $10D$ AND $20D$

Algorithm	$10D$	$20D$	Total
NL-SHADE-RSP	7+/3=/2-	6+/4=/2-	13+/7=/4-
APGSK-IMODE	10+/1=/1-	8+/1=/3-	18+/2=/4-
MLS-LSHADE	9+/1=/2-	5+/3=/4-	14+/4=/6-
MadDE	11+/0=/1-	7+/3=/2-	18+/3=/3-
L-SHADE-RSP	7+/4=/1-	4+/5=/3-	11+/9=/4-

The comparison in Table I shows that NL-SHADE-LBC is better than the alternative methods, including the NL-SHADE-RSP algorithm in both $10D$ and $20D$ scenarios. However, when the comparison is performed not pairwise, but between a set of approaches, the efficiency of NL-SHADE-LBC is not as high. Table II contains the Friedman ranking of NL-SHADE-LBC against other algorithms participating in the comparison.

As can be seen from Table II, in $10D$ scenario the NL-SHADE-LBC shows superior results compared to all methods, having a total rank of 11.1, but for $20D$ functions the results are not as good. In particular, the MLS-LSHADE algorithm, which applies the local search using Davies, Swann, and Campey method with Gram-Schmidt orthogonalization (DSCG) to the best solutions is able to get a lower rank

TABLE II
FRIEDMAN RANKING OF NL-SHADE-LBC AND OTHER APPROACHES, $10D$ AND $20D$

Algorithm	$10D$	$20D$	Total
NL-SHADE-RSP	33.62	38.37	71.98
APGSK-IMODE	36.13	39.33	75.47
MLS-LSHADE	29.08	19.53	48.62
MadDE	39.72	38.43	78.15
L-SHADE-RSP	30.35	22.78	53.13
NL-SHADE-LBC	11.10	21.55	32.65

of 19.53 compared to 21.55, which NL-SHADE-LBC got. This happens because although in pairwise comparison NL-SHADE-LBC is often better, when comparing a set of methods, due to poor results on several functions, NL-SHADE-LBC gets much worse ranks.

Tables III and IV contain the results of pairwise Mann-Whitney statistical tests against other algorithms for $10D$ and $20D$ respectively. The values in the cells are the test result (1 means that NL-SHADE-LBC is significantly better, -1 means that it was significantly worse, 0 means that difference was insignificant), and standard score returned from the normally approximated Mann-Whitney test (Z value). The threshold values for Z were set to -2.58 and 2.58.

TABLE III
MANN-WHITNEY TESTS OF NL-SHADE-LBC AGAINST OTHER APPROACHES FOR EVERY FUNCTION, TEST RESULT AND STANDARD SCORE, $10D$

Algorithm	F1	F2	F3	F4
NL-SHADE-RSP	1, 6.65	1, 6.21	1, 6.65	1, 6.66
APGSK-IMODE	1, 6.66	1, 6.21	1, 6.65	1, 5.79
MLS-LSHADE	1, 6.65	1, 6.33	1, 6.65	1, 4.38
MadDE	1, 6.65	1, 6.21	1, 6.65	1, 6.63
L-SHADE-RSP	1, 6.65	1, 6.63	1, 6.65	0, 1.73
Algorithm	F5	F6	F7	F8
NL-SHADE-RSP	1, 6.65	0, 2.56	1, 5.12	0, 2.07
APGSK-IMODE	1, 6.65	0, 2.26	1, 6.65	1, 4.95
MLS-LSHADE	1, 6.65	0, 2.53	1, 6.65	1, 6.65
MadDE	1, 6.65	1, 4.05	1, 6.65	1, 4.21
L-SHADE-RSP	1, 6.65	0, 2.00	1, 6.65	1, 6.49
Algorithm	F9	F10	F11	F12
NL-SHADE-RSP	0, 0.00	-1, -6.66	1, 6.65	-1, -5.64
APGSK-IMODE	1, 7.68	1, 6.34	1, 6.65	-1, -6.86
MLS-LSHADE	1, 6.01	-1, -4.88	1, 6.65	-1, -6.97
MadDE	1, 7.68	1, 6.22	1, 6.65	-1, -7.03
L-SHADE-RSP	0, 0.00	0, 0.34	1, 6.65	-1, -4.43

According to the comparison in Tables III and IV, it can be noted that NL-SHADE-LBC performs worse than other methods on certain functions, namely F2, Shifted and Rotated Rosenbrock's Function, and F10, which is a composition Function combining Schwefel's Function, Rastrigin's Function and HGBat Function. The poor results of NL-SHADE-LBC on F2 are the main reason of losing to MLS-SHADE in ranking in $20D$.

The complexity of the NL-SHADE-LBC algorithm was estimated by calculating T0, T1 and T2 values according to [5]. The results are presented in Table V.

TABLE IV
MANN-WHITNEY TESTS OF NL-SHADE-LBC AGAINST OTHER APPROACHES FOR EVERY FUNCTION, TEST RESULT AND STANDARD SCORE, $20D$

Algorithm	F1	F2	F3	F4
NL-SHADE-RSP	1, 6.65	-1, -4.58	1, 6.65	1, 6.65
APGSK-IMODE	1, 6.66	-1, -6.55	1, 6.65	1, 6.65
MLS-LSHADE	1, 3.24	-1, -6.70	1, 6.65	0, 2.14
MadDE	1, 6.65	-1, -6.56	1, 6.65	1, 6.65
L-SHADE-RSP	1, 6.65	-1, -5.22	1, 6.65	0, 0.88
Algorithm	F5	F6	F7	F8
NL-SHADE-RSP	1, 6.65	1, 5.84	1, 5.32	0, -1.44
APGSK-IMODE	1, 6.65	1, 6.61	1, 5.68	0, 1.21
MLS-LSHADE	1, 6.65	0, -0.89	1, 4.64	0, -1.70
MadDE	1, 6.65	0, -0.98	1, 4.70	0, -1.11
L-SHADE-RSP	1, 6.65	0, 2.13	0, 0.52	-1, -3.34
Algorithm	F9	F10	F11	F12
NL-SHADE-RSP	0, 0.00	-1, -6.65	0, -1.00	0, -0.24
APGSK-IMODE	1, 7.68	1, 6.58	-1, -3.88	-1, -5.22
MLS-LSHADE	1, 7.68	-1, -5.77	-1, -6.35	-1, -5.19
MadDE	1, 7.68	1, 6.65	0, -1.00	-1, -6.26
L-SHADE-RSP	0, 0.00	1, 4.66	0, 0.59	-1, -5.23

TABLE V
COMPUTATIONAL COMPLEXITY OF NL-SHADE-LBC

D	$T0$	$T1$	$T2$	$(T2-T1)/T0$
$D=10$	2.6E-5	2.4E-5	1.498E-2	4.83864
$D=20$	2.6E-5	6.1E-5	2.818E-2	8.49231

Complexity measurement results, presented in Table V, show that NL-SHADE-LBC scales linearly with problem dimension.

To allow the comparison to other approaches, Tables VI-VII contain the best, worst, mean, median and standard deviation values achieved by NL-SHADE-LBC on all functions.

TABLE VI
NL-SHADE-LBC RESULTS, $10D$

Function	Best	Worst	Median	Mean	Std
f1	0.0	0.0	0.0	0.0	0.0
f2	0.0	0.0	0.0	1.33E-01	7.16E-01
f3	0.0	0.0	0.0	0.0	0.0
f4	2.16E-05	2.99E+00	9.95E-01	1.30E+00	7.78E-01
f5	0.0	0.0	0.0	0.0	0.0
f6	4.31E-03	4.40E-01	7.21E-02	1.24E-01	1.25E-01
f7	0.0	0.0	0.0	0.0	0.0
f8	2.96E-04	1.79E-01	3.29E-02	4.60E-02	3.80E-02
f9	2.29E+02	2.29E+02	2.29E+02	2.29E+02	5.68E-14
f10	1.00E+02	1.00E+02	1.00E+02	1.00E+02	2.95E-02
f11	0.0	0.0	0.0	0.0	0.0
f12	1.63E+02	1.65E+02	1.65E+02	1.65E+02	4.04E-01

Figures 2 and 3 show the convergence graphs of all the methods, which participated in the comparison.

V. CONCLUSIONS

In this paper the NL-SHADE-LBC algorithm was introduced, which is based on NL-SHADE-RSP and includes several novel features. The NL-SHADE-LBC introduces linear bias change in parameter adaptation, as well as solution

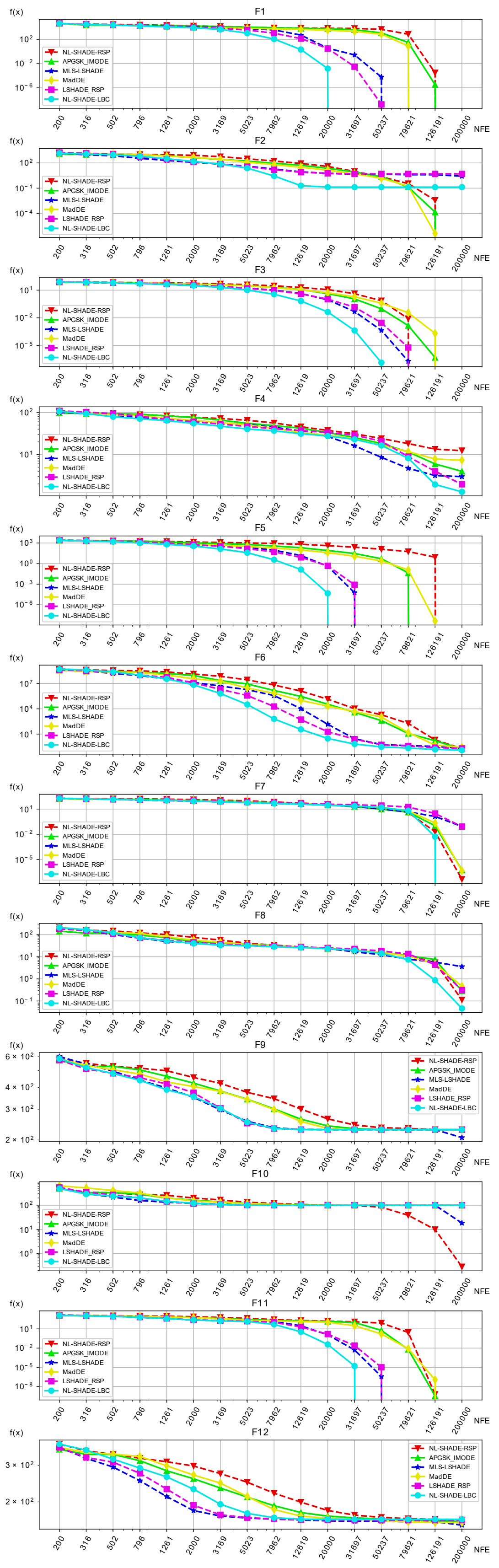

Fig. 2. Convergence graphs of all tested algorithms, $10D$

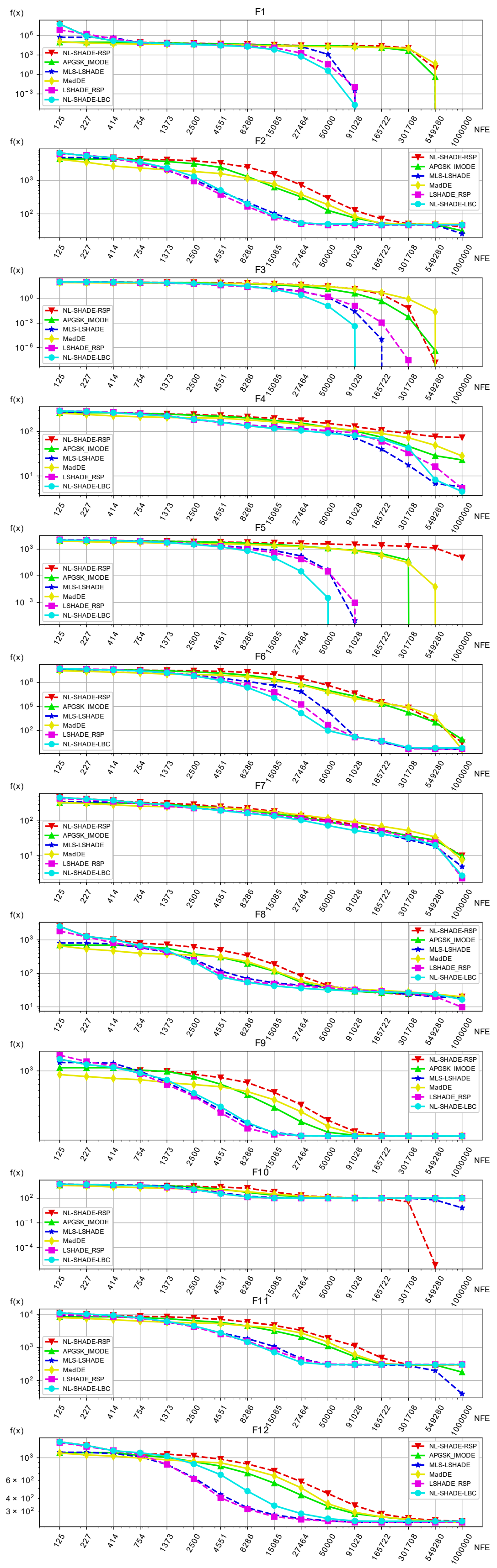

Fig. 3. Convergence graphs of all tested algorithms, $20D$

TABLE VII
NL-SHADE-LBC RESULTS, $20D$

Function	Best	Worst	Median	Mean	Std
f1	0.0	0.0	0.0	0.0	0.0
f2	8.01E-03	4.91E+01	4.91E+01	4.73E+01	8.82E+00
f3	0.0	0.0	0.0	0.0	0.0
f4	1.99E+00	7.96E+00	4.97E+00	4.45E+00	1.40E+00
f5	0.0	0.0	0.0	0.0	0.0
f6	7.18E-02	2.48E+00	4.64E-01	6.36E-01	5.60E-01
f7	1.18E-03	2.01E+01	4.51E-01	2.58E+00	5.74E+00
f8	3.44E-01	2.11E+01	2.05E+01	1.65E+01	6.33E+00
f9	1.81E+02	1.81E+02	1.81E+02	1.81E+02	0.0
f10	1.00E+02	1.00E+02	1.00E+02	1.00E+02	2.29E-02
f11	3.00E+02	4.00E+02	3.00E+02	3.03E+02	1.80E+01
f12	2.30E+02	2.48E+02	2.39E+02	2.39E+02	4.13E+00

resampling, and uses exponential rank-based selection as in NL-SHADE-RSP with different coefficient. The performed computational experiments have shown that NL-SHADE-LBC is able to achieve better results than alternative approaches on CEC 2022 benchmark functions, which makes it highly competitive approach.

REFERENCES

[1] Antoniou, A., Lu, W.S., ”Practical Optimization: Algorithms and Engineering Applications”, Practical Optimization, 2021, doi: 10.1007%2F978-1-0716-0843-2.

[2] 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.

[3] 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.

[4] 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.

[5] Kumar A., Price K.V., Mohamed A.W., Hadi A.A., P. N. Suganthan, ”Problem Definitions and Evaluation Criteria for the CEC 2022 Special Session and Competition on Single Objective Bound Constrained Numerical Optimization”, Technical Report, Nanyang Technological University, Singapore, December 2021.

[6] V. Stanovov, S. Akhmedova and E. Semenkin, ”NL-SHADE-RSP algorithm with adaptive archive and selective pressure for CEC 2021 numerical optimization,” 2021 IEEE Congress on Evolutionary Computation (CEC), 2021, pp. 809-816, doi: 10.1109/CEC45853.2021.9504959.

[7] Mohamed A.W., Hadi A.A., Mohamed A.K., Agrawal P., Kumar A., P. N. Suganthan, ”Problem Definitions and Evaluation Criteria for the CEC 2021 Special Session and Competition on Single Objective Bound Constrained Numerical Optimization”, Technical Report, Nanyang Technological University, Singapore.

[8] Cuong L.V., Bao, N.N., Binh H.T., ”A Multi-start Local Search Algorithm with L-SHADE for Single Objective Bound Constrained Optimization”, Technical report, Hanoi University of Science and Technology, Vietnam, 2021.

[9] Biswas, S., Saha, D., De, S., Cobb, A.D., Das, S., and Jalaian, B., ”Improving Differential Evolution through Bayesian Hyperparameter Optimization”, 2021 IEEE Congress on Evolutionary Computation (CEC), 2021, pp. 832-840, doi: 10.1109/CEC45853.2021.9504792.

[10] Mohamed A.W., Hadi A.A., Agrawal P., Sallam K.M., and Mohamed A.K., ”Gaining-Sharing Knowledge Based Algorithm with Adaptive Parameters Hybrid with IMODE Algorithm for Solving CEC 2021 Benchmark Problems”, 2021 IEEE Congress on Evolutionary Computation (CEC), 2021, pp. 841-848, doi: 10.1109/CEC45853.2021.9504814.

[11] 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.

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

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

[14] 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).

[15] 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.

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

[17] Mohamed, A. W., A. A. Hadi, A. Mohamed, Noor H. Awad. “Evaluating the Performance of Adaptive GainingSharing Knowledge Based Algorithm on CEC 2020 Benchmark Problems.” 2020 IEEE Congress on Evolutionary Computation (CEC) (2020): 1-8.

[18] Sallam, K. M., S. Elsayed, R. K. Chakrabortty, M. Ryan. “Improved Multi-operator Differential Evolution Algorithm for Solving Unconstrained Problems.” 2020 IEEE Congress on Evolutionary Computation (CEC) (2020): 1-8.

[19] 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.

[20] V. Stanovov, S. Akhmedova and E. Semenkin, ”Biased parameter adaptation in differential evolution”, Information Sciences, Vol. 566, pp. 215-238, 2021, doi: 10.1016/J.INS.2021.03.016.

[21] 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.

[22] V. Stanovov, S. Akhmedova, E. Semenkin and M. Semenkina, ”Generalized Lehmer Mean for Success History based Adaptive Differential Evolution”, In Proceedings of the 11th International Joint Conference on Computational Intelligence - ECTA, IJCCI 2019, pp. 93-100, doi: 10.5220/0008163600930100.

[23] P.S. Bullen, ”Handbook of Means and Their Inequalities”, Springer Netherlands, 2003, doi: 10.1007/978-94-017-0399-4.

[24] V. Stanovov, S. Akhmedova and E. Semenkin, ”Archive Update Strategy Influences Differential Evolution Performance”, Advances in Swarm Intelligence, 2020, 12145, pp. 397 - 404, doi: 10.1007/978-3-030-53956-6_35.