mathematics

MDPI

Article

Dual-Population Adaptive Differential Evolution Algorithm L-NTADE

Vladimir Stanovov [1,2,*], Shakhnaz Akhmedova [3] and Eugene Semenkin [1,2]

[1] School of Space and Information Technologies, Siberian Federal University, 660074 Krasnoyarsk, Russia
[2] Institute of Informatics and Telecommunication, Reshetnev Siberian State University of Science and Technology, 660037 Krasnoyarsk, Russia
[3] Independent Researcher, 12489 Berlin, Germany
* Correspondence: vladimirstanovov@yandex.ru

Abstract: This study proposes a dual-population algorithmic scheme for differential evolution and specific mutation strategy. The first population contains the newest individuals, and is continuously updated, whereas the other keeps the top individuals throughout the whole search process. The proposed mutation strategy combines information from both populations. The proposed L-NTADE algorithm (Linear population size reduction Newest and Top Adaptive Differential Evolution) follows the L-SHADE approach by utilizing its parameter adaptation scheme and linear population size reduction. The L-NTADE is tested on two benchmark sets, namely CEC 2017 and CEC 2022, and demonstrates highly competitive results compared to the state-of-the-art methods. The deeper analysis of the results shows that it displays different properties compared to known DE schemes. The simplicity of L-NTADE coupled with its high efficiency make it a promising approach.

Keywords: differential evolution; population size; parameter adaptation

MSC: 65K10; 68W50

Citation: Stanovov, V.; Akhmedova, S.; Semenkin, E. Dual-Population Adaptive Differential Evolution Algorithm L-NTADE. *Mathematics* **2022**, *10*, 4666. https://doi.org/10.3390/math10244666

Academic Editors: Yu Xue, Chunlin He and Ferrante Neri

Received: 21 November 2022
Accepted: 6 December 2022
Published: 9 December 2022

1. Introduction

Currently, the area of evolutionary algorithms (EA) is rapidly developing along with other computational intelligence methods (CI) methods, such as neural networks (NN) and fuzzy logic systems (FL). The heuristic optimization approaches proposed within EA and swarm intelligence (SI) frameworks are aimed at finding the best possible algorithmic schemes capable of solving complex global optimization problems [1]. Specific versions of algorithms are developed for constrained, multi-objective, many-objective, Boolean, integer and bilevel optimization [2]. Nevertheless, the algorithms proposed for single-objective numerical problems often serve as a fundament for other directions of studies and are often applied for solving complex engineering problems [3,4].

In recent years, differential evolution (DE) [5] has attracted the attention of many researchers as, unlike other EA and swarm intelligence methods, such as genetic algorithms (GA) [6], evolutionary strategies (ES) [7], particle swarm optimization (PSO) [8] and many others [9,10], it is characterized by high efficiency and simplicity in implementation. This is reflected in the number of participants in the recent optimization competitions, such as the IEEE Congress on Evolutionary Computation (CEC), where most of the submitted algorithms and winners are DE-based methods [11].

The studies on differential evolution are mainly concentrated on the problem of parameter adaptation as DE is known to be highly sensitive [12,13] to the three main parameters, namely scaling factor, crossover rate and population size. Various adaptation schemes were proposed, starting with the SaDE [14] algorithm, where the scaling factor was sampled with normal distribution, while crossover rate was learned based on experience. Other approaches, such as [15–17], used a predefined pool of parameter values. A relatively

simple randomization of parameter values has been shown to perform well, as jDE [18] has demonstrated, followed by similar approaches. The development of the JADE algorithm [19], where memory cells were used to store successful values, was followed by SHADE [20] as the most popular and the L-SHADE [21] with population size reduction, as well as many others, such as [22]. Recent studies on differential evolution have resulted in many approaches, such as TVDE (with time-varying strategy) [23], CSDE (with combined mutation strategies) [24], qlDE (with Q-learning based parameter tuning strategy) [25], MPPCEDE (with multi-population and multi-strategy) [26] and RL-HPSDE (with adaptation based on reinforcement learning) [27]. Attempts have been also made to realize the automatic design of parameter adaptation in DE using genetic programming [28] and neuroevolution [29].

However, the main algorithmic scheme of DE remains the same. In most studies, it has a single population and an optional external archive, and the replacement occurs only if an offspring is better than a parent. In some studies, there have been attempts to deviate from the prevalent schemes by introducing hierarchical archives in HARD-DE [30], big and small populations in j21 [31], junior and senior individuals in a DE-like AGSK [32], and global replacement in GRDE [33]. Recently, the Unbounded DE (UDE) has been proposed in [34], where the population may infinitely grow, and specific selection mechanisms are applied to drive the search.

In this study, we further develop the ideas of UDE and propose a two-population DE algorithm, with the first population called newest and the second population called top. The newest population has a specific update rule, keeping the last good solutions, and the top population keeping best found solutions during the entire search. The resulting L-NTADE algorithm (Linear population size reduction Newest and Top Adaptive Differential Evolution) is considered in several modifications with various mutation strategies. The algorithm is tested on the CEC 2017 [35] and CEC 2022 [36] benchmark sets and demonstrates high efficiency and specific properties on some of the functions. The main features of this study can be outlined as follows:

1. The new dual-population DE scheme with a version of the *current-to-pbest* mutation strategy using individuals from the top population as one of the p% best is superior compared to other strategies;
2. The new selection (replacement) rule for the newest population allows the algorithm to significantly improve performance, compared to the case when classical selection is used;
3. The proposed L-NTADE algorithm performs better on complex multimodal test problems.

The Section 2 contains an overview of related work, the Section 3 describes the proposed approach, the Section 4 contains the experimental setup and results, then a discussion of the results is provided, and the Section 5 concludes the paper.

2. Related Work

2.1. Differential Evolution

Differential evolution is a popular heuristic numerical optimization method, originally proposed by Storn and Price [37]. DE is a population-based method, so it starts by randomly initializing a set of N individuals $x_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,D}), i = 1, \ldots, N$ within the search range:

$$S = \{x_i \in R^D | x_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,D}) : x_{i,j} \in [x_{lb,j}, x_{ub,j}]\} \quad (1)$$

where $j = 1, \ldots, D$ and D is the search space dimensionality. Each individual is generated using uniform distribution:

$$x_{i,j} = x_{lb,j} + rand \times (x_{ub,j} - x_{lb,j}). \quad (2)$$

Although DE was proposed for numerical single-objective unconstrained problems, it can by modified for other types of problems [12]. The main feature of DE is the difference-

based mutation operator, which is a key component of the search process. There exist several variants of mutation strategies, including *rand/1*, *rand/2*, *best/1*, *best/2*, *current-to-best/1* and *current-to-pbest/1* [13]. The original version, *rand/1*, generates new a solution as follows:

$$v_{i,j} = x_{r1,j} + F \times (x_{r2,j} - x_{r3,j}), \tag{3}$$

where v_i is called the mutant or donor vector, $x_{i,j}$, is the j-th coordinate of the i-th candidate solution, the indexes i, $r1$, $r2$ and $r3$ are all mutually different, and F is the scaling factor chosen from $[0, 2]$. The scaling factor parameter is one of the most important for DE as the algorithm was shown to be highly sensitive to its values [5].

After the mutation, the crossover step is performed, which combines the generated donor vector and the target vector used as a baseline in the mutation, i.e., the i-th individual in the population. The resulting trial vector u_i is usually generated with a binomial crossover operator:

$$u_{i,j} = \begin{cases} v_{i,j}, & \text{if } rand(0,1) < Cr \text{ or } j = jrand \\ x_{i,j}, & \text{otherwise} \end{cases}. \tag{4}$$

In this formula, $Cr \in [0, 1]$ is the crossover rate, and *jrand* is a randomly chosen index from $[1, D]$. The *jrand* index is required to make sure that at least one component is inherited from the donor vector. Otherwise, evaluating a copy of an individual would be a waste of computational resources. A recent study has shown that despite this fix, the problem of duplicate individuals may still occur in DE [38].

Applying a mutation operator may result in solutions that are outside of the search space boundaries. Hence, a specific bound constraint handling method (BCHM) should be applied in DE. In particular, a popular method for this is called midpoint-target, where each j-th ($j = 1, \ldots, D$) coordinate of the i-th ($i = 1, \ldots, N$) vector is returned to the interval $[x_{lb,j}, x_{ub,j}]$ as follows:

$$u_{i,j} = \begin{cases} \frac{x_{lb,j}+x_{i,j}}{2}, & \text{if } v_{i,j} < x_{lb,j} \\ \frac{x_{ub,j}+x_{i,j}}{2}, & \text{if } v_{i,j} > x_{ub,j} \end{cases}. \tag{5}$$

Here, if the j-th component of the mutant vector is greater than the upper boundary or smaller than the lower boundary of the corresponding interval $[x_{lb,j}, x_{ub,j}]$, then its parent vector x_i is used to set the new value for this component. Note that this step can be applied after mutation or after crossover.

The last step in the classical DE scheme is called selection, but unlike selection in a genetic algorithm, it plays the role of a replacement operator. If the newly generated trial vector u_i is better than the corresponding target vector, then the replacement occurs:

$$x_i = \begin{cases} u_i, & \text{if } f(u_i) \leq f(x_i) \\ x_i, & \text{if } f(u_i) > f(x_i) \end{cases}. \tag{6}$$

Although this selection mechanism is known to be simple and efficient, there have been some attempts to improve it, for example, by using the information about neighborhoods [39].

2.2. DE Modifications

Due to the high popularity of DE variants on evolutionary computation and the huge number of studies, a comprehensive survey of all existing methods here would be impractical. Interested readers are therefore advised to refer to surveys such as [12,13,40], specialized studies about certain types of DE, for example [22] or operators [41], as well as some of our previous studies on selective pressure [42] and parameter adaptation [43]. Nevertheless, here we will focus on some studies that are of particular interest for the current work.

One of the important milestones of DE development was the JADE algorithm, proposed by Zhang and Sanderson [44]. JADE introduced one of the most efficient mutation

strategies *current-to-pbest/1*, which is used in most DE variants today, and can be described as follows:

$$v_{i,j} = x_{i,j} + F \times (x_{pbest,j} - x_{i,j}) + F \times (x_{r1,j} - x_{r2,j}), \quad (7)$$

where *pbest* is the index of one of the $pb * 100\%$ best individuals, different from i, $r1$ and $r2$. The two brackets containing differences implement two main features, namely exploitation by moving towards one of the best solutions, and exploitation by adding a difference vector between two randomly chosen solutions. Moreover, increasing F to 1 means generating solutions closer to the best and at the same time making a larger step with a second difference, while smaller F values mean exploration close to the target vector. JADE has also introduced the concept of an external archive A, a set of solutions that were replaced by better ones during selection. The solutions from the archive are used in *current-to-pbest/1* instead of the last vector x_{r2}. The archive was shown to be of major importance for improving the search efficiency, and archive handling techniques are an important field of studies [30,45].

The efficiency of JADE has inspired other researchers to develop its improved versions. In particular, the SHADE algorithm proposed by Tanabe and Fukunaga [20] has improved the parameter adaptation of JADE by introducing a set of H memory cells $(M_{F,h}, M_{Cr,h})$, each containing a couple of F and Cr values. For every mutation and crossover operator, the parameter values are sampled as follows:

$$\begin{cases} F = randc(M_{F,k}, 0.1) \\ Cr = randn(M_{Cr,k}, 0.1) \end{cases}. \quad (8)$$

Here, *randc* is a Cauchy distributed random value, *randn* is a normally distributed random number, and k is chosen from $[1, H]$ for each individual. If the generated Cr value is outside the $[0, 1]$ range, it is truncated to this range. If F is larger than 1, it is set to 1, and if F is smaller than 0, it is generated again until it becomes positive. At the end of every generation, the memory cell with the index h (iterated from 1 to H every generation) is updated using the successful F and Cr values. The successful parameter values are the ones which delivered an improvement in terms of fitness, i.e., if an offspring replaced a parent, then F and Cr are stored in the S_F and S_{Cr} arrays, and the improvement value $\Delta f = |f(u_j) - f(x_j)|$ is stored in $S_{\Delta f}$. The update of the memory cell is performed by first calculating the weighted Lehmer mean [46]:

$$mean_{wL} = \frac{\sum_{j=1}^{|S|} w_j S_j^2}{\sum_{j=1}^{|S|} w_j S_j}, \quad (9)$$

where $w_j = \frac{S_{\Delta f_j}}{\sum_{k=1}^{|S|} S_{\Delta f_k}}$, S is either S_{Cr} or S_F.

The values in the memory cell are updated as follows:

$$\begin{cases} M_{F,k}^{t+1} = 0.5(M_{F,k}^{t} + mean_{(wL,F)}) \\ M_{Cr,k}^{t+1} = 0.5(M_{Cr,k}^{t} + mean_{(wL,Cr)}) \end{cases}, \quad (10)$$

where t is the current iteration number.

In [43], the biased parameter adaptation was proposed, modifying the Lehmer mean by introducing an additional parameter pm:

$$mean_{wL} = \frac{\sum_{j=1}^{|S|} w_j S_j^{pm}}{\sum_{j=1}^{|S|} w_j S_j^{pm-1}}, \quad (11)$$

The additional parameter allows the adaptation of either F or Cr to be skewed towards smaller or larger values. The standard setting in L-SHADE is $pm = 2$, and in [43], it is

shown that increasing this value and generating a larger F may lead to much better results in high-dimensional problems.

Another important modification of SHADE was L-SHADE [21], which introduced a simple control strategy for population size, called Linear Population Size Reduction (LPSR). The algorithm starts with NP_{max} individuals in the population, and gradually reduces their number to NP_{min} individuals:

$$NP_{g+1} = round\left(\frac{NP_{min} - NP_{max}}{NFE_{max}} NFE\right) + NP_{max}, \tag{12}$$

where $NP_{min} = 4$, NFE and NFE_{max} are the current and total number of available function evaluations, respectively. At the end of every generation, the worst solutions are removed from the population if it is required, and the archive size is also decreased. Some recent studies have proposed L-SHADE variants with a very large population initialized by orthogonal design [47].

In [42], the effects of selective pressure on DE performance were studied, and it was shown that adding tournament or rank-based selection strategies may be beneficial. The exponential rank-based selection was implemented by selecting an individual depending on its fitness in a sorted array, with the ranks assigned as follows:

$$rank_i = e^{\frac{-kp \cdot i}{NP}}, \tag{13}$$

where kp is the parameter controlling the pressure, and i is the individual number. Larger ranks are assigned to better individuals, and a discrete distribution is used for selection.

The importance of the L-SHADE algorithm is proved by the number of modifications proposed for it. For example, jSO [48] proposed a modified mutation strategy and specific rules for parameter adaptation depending on the stage of the search, L-SHADE-RSP [49] introduced rank-based selective pressure, LSHADE-SPACMA [50] used a hybridization with CMA-ES, NL-SHADE-RSP [51] proposed non-linear population size reduction, crossover rate sorting and adaptive archive usage, MLS-LSHADE [52] added multi-start local search, and DB-LSHADE proposed distance-based parameter adaptation [53]. Although all these studies have shown different possibilities of modern DE methods, according to a recent study on Unbounded DE [34], "The notion of a population with individuals which are replaced by newly generated individuals is a pervasive idea in differential evolution". In the next section, an algorithmic scheme that deviates from this concept is proposed.

3. Proposed Approach

Inspired by the experiments with unbounded population in UDE and supported by several preliminary tests with this setup, the L-NTADE algorithm is proposed. The L-NTADE maintains two populations, the first called the newest population, and the second named the top population. Unlike UDE, both populations are limited in size as the preliminary tests have shown that handling very large populations requires significant computational efforts.

The L-NTADE algorithm starts by initializing a population of N_{max} individuals x_i^{new}, $i = 1, \ldots, N_{max}$. After that, the individuals in this population are copied to the top population x^{top}.

The L-NTADE uses variants of the *current-to-pbest* mutation strategy and the parameter adaptation from SHADE algorithm, but without an external archive. The mutation strategies considered in this study are the following:

1. *r-new-to-ptop/t/t*: $v_{i,j} = x_{r1,j}^{new} + F \times (x_{pbest,j}^{top} - x_{i,j}^{new}) + F \times (x_{r2,j}^{top} - x_{r3,j}^{top})$,
2. *r-new-to-ptop/t/n*: $v_{i,j} = x_{r1,j}^{new} + F \times (x_{pbest,j}^{top} - x_{i,j}^{new}) + F \times (x_{r2,j}^{top} - x_{r3,j}^{new})$,
3. *r-new-to-ptop/n/t*: $v_{i,j} = x_{r1,j}^{new} + F \times (x_{pbest,j}^{top} - x_{i,j}^{new}) + F \times (x_{r2,j}^{new} - x_{r3,j}^{top})$,
4. *r-new-to-ptop/n/n*: $v_{i,j} = x_{r1,j}^{new} + F \times (x_{pbest,j}^{top} - x_{i,j}^{new}) + F \times (x_{r2,j}^{new} - x_{r3,j}^{new})$.

Here, the following notation is used. The terms *r-new* and *r-top* stand for the choice of a random individual from the newest population or top population as a target solution, *pnew* and *ptop* for the choice of one of the *pb*% best individuals in the newest or top populations, and */t/n* indicates the usage of individuals from either the top or new population in the second difference. Note that the target vector is not the *i*-th vector as in most DE, but a randomly chosen vector from one of the populations. These mutation strategies were chosen as they represent different scenarios of applying individuals from one of the populations, and here the two last variants are the extreme cases, when only one of the populations is used, and the others are intermediate. The number of possible combinations here is significant, and we only consider the cases for which the efficiency level is unclear. Additionally, note that as the indexes are chosen from different populations, equal indexes should be checked only if they are from the same population.

The control of the *pb* parameter is performed in the following way. At the beginning of the search, *pb* is set to pb_{max} and it is linearly reduced down to pb_{min}:

$$pb_{g+1} = round(\frac{pb_{min} - pb_{max}}{NFE_{max}} NFE) + pb_{max} \tag{14}$$

where g is the current generation number, and will be omitted further for simplicity. Additionally, if the number of best individuals to choose from is less than 2, then it is set to 2. The linear decrease of *pb* during the search leads to increased greediness closer to the end of the search.

The crossover step in L-NTADE is unchanged, i.e., the classical binomial crossover is used to generate u_i as in the L-SHADE algorithm. The bound constraint handling method used is the midpoint target, described in the previous subsection.

The selection step, however, is one of the main features of the L-NTADE algorithm. The main idea here is to imitate the behavior of the unbounded population from which the newest and best individuals are chosen by maintaining two populations. The selection step depends on the mutation strategy, in particular, if the target solution was chosen from the top or newest population. The main idea for an update is still the same: if the trial vector is better than the target, it should be saved. However, the new solutions are always saved to the newest population. The index of an individual *nc* to which the trial vector is copied is iterated from 1 to N_{cur} after every successful solution, and reset to 1 once it reaches N_{cur}. The selection step can be described as follows:

$$x_{nc} = \begin{cases} u_i, & \text{if } f(u_i) \leq f(x_{r1}^{t|n}) \\ x_{nc}, & \text{if } f(u_i) > f(x_{r1}^{t|n}) \end{cases}. \tag{15}$$

Here, $t|n$ means that the target vector can be chosen from either the top or newest population according to the used mutation strategy. The successful trial vectors are copied to the current newest population immediately, and due to the random choice of solutions for mutation, can be used for generating other vectors within the same generation. Although the choice of the *pb*% best individuals' indexes is performed only once a generation, we believe that there is not much sense in sorting and finding best solutions after every successful selection. Moreover, this could possibly be a problem only for the 5th mutation variant. All successful solutions are additionally stored in a temporary pool x^{temp}, and at the end of the generation x^{top} and x^{temp} are joined, sorted and only N_{cur} individuals are saved to x^{top}, where N_{cur} is the current size of both populations. In this way, the top population always contains N_{cur} individuals from the whole search.

The population size control strategy is the same as in L-SHADE, with the only difference being that the newest and top populations are both linearly decreasing their size, and with the same initial and final size. The following equation is true for both populations, with N_{max} and N_{min}:

$$N_{cur}^{g} = round(\frac{N_{min} - N_{max}}{NFE_{max}} NFE) + N_{max} \tag{16}$$

The pseudocode of the L-NTADE algorithm is shown in Algorithm 1.

Algorithm 1 L-NTADE

1: Input: D, NFE_{max}, N_{max}, goal function $f(x)$
2: Output: x_{best}^{top}, $f(x_{best}^{top})$
3: Set $N_{cur}^{0} = N_{max}$, $N_{min} = 4$, $H = 5$, $M_{F,r} = 0.3$, $M_{Cr,r} = 1$
4: Set $pb = 0.3$, $k = 1$, $g = 0$, $nc = 1$, $kp = 0$, $pm = 2$
5: Initialize population $(x_{1,j}^{new}, ..., x_{N_{max},j}^{new})$ randomly, calculate $f(x^{new})$
6: Copy x^{new} to x^{top}, $f(x^{new})$ to $f(x^{top})$ $NFE < NFE_{max}$
7: $S_F = \emptyset$, $S_{Cr} = \emptyset$, $S_{\Delta f} = \emptyset$
8: Rank either x^{new} by $f(x^{new})$ $i = 1$ to N_{cur}^{g}
9: $r1 = randInt(N_{cur}^{g})$
10: Current memory index $r = randInt[1, H+1]$
11: Crossover rate $Cr_i = randn(M_{Cr,r}, 0.1)$
12: $Cr_i = min(1, max(0, Cr))$
13: $F_i = randc(M_{F,r}, 0.1)$ $F_i \geq 0$
14: $F_i = min(1, F_i)$
15: $pbest = randInt(1, N_{cur}^{g} \cdot pb)$
16: $r2 = randInt(1, N_{cur}^{g})$ or with rank-based selection
17: $r3 = randInt(1, N_{cur}^{g})$ indexes $r1$, $r2$, $r3$ and $pbest$ are different
18: Apply mutation to produce v_i with F_i
19: Apply binomial crossover to produce u_i with Cr_i
20: Apply bound constraint handling method
21: Calculate $f(u_i)$ $f(u_i) < f(x_{r1}^{new})$
22: $u_i \rightarrow x^{temp}$
23: $F \rightarrow S_F$, $Cr \rightarrow S_{Cr}$
24: $\Delta f = f(x_{r1}^{new}) - f(u_i)$
25: $\Delta f \rightarrow S_{\Delta f}$
26: $x_{nc}^{new} = u_i$
27: $nc = mod(nc + 1, N_{cur}^{g})$
28: Get N_{cur}^{g+1} with LPSR
29: Join together x^{top} and x^{temp}, sort and copy best N_{cur}^{g+1} to x^{top} $N_{cur}^{g} > N_{cur}^{g+1}$
30: Remove worst individuals from x^{new}
31: Update $M_{F,k}$, $M_{Cr,k}$
32: $k = mod(k + 1, H)$
33: $g = g + 1$
34: Return x_{best}^{top}, $f(x_{best}^{top})$

The algorithm requires a goal function, problem dimension, total computational resource and initial population size to run, and returns the best solution along with its value, as the first two lines of Algorithm 1 show. After this, the main parameters are set in lines 3 and 4. Line 5 describes the initialization step, where the newest population is filled with random individuals. In line 6, the new population is copied to the top population, together with goal function values. Next, the main loop is started, where at the beginning of each generation in line 8, the sets of successful F, Cr and Δf values are emptied. In line 9, the population of new individuals is sorted according to the fitness values, and the ranks are assigned to the individuals. After this, the loop over individuals is started in line 10. As the mutation strategy requires random indexes, in line 11 the first of them is generated. Next, the current memory index is randomly chosen in line 12, and this index is used to generate Cr and F values in lines 13–14 and 15–18, respectively. In lines 19–23, the rest of the indexes for mutation are generated until they are mutually different. In line 21, the $r2$ index can be generated randomly or by using ranks calculated in line 9 depending on the mutation strategy used. Lines 24, 25 and 26 implement the main search operators of DE, such as mutation and crossover, as well as a bound-constraint handling method.

After this, once the trial vector is generated, its fitness value is calculated in line 27. In lines 28–35, the selection is performed, i.e., if the trial vector is better than the randomly chosen one with index $r1$, then it is saved in the temporary population, and the current F and Cr values are saved together with Δf. Additionally, in line 33, one of the individuals from the new population is replaced by a trial vector, and the index of the individual to be replaced is updated in line 34. Line 36 finishes the loop over individuals, and in line 37 the population size is updated according to LPSR. Before shrinking the populations in lines 39–41, the top population and temporary population are joined together, sorted so that the best individuals are saved in the top population in line 38. At the end of the generation, in line 42 the memory cells are updated using S_F, S_{Cr} and Δf, in line 43 the memory cell index to be updated is incremented, and the generation number in line 44 does the same. Finally, line 45 finishes the main loop over function evaluations, and line 45 returns the result.

The flow chart of the L-NTADE algorithm is shown in Figure 1.

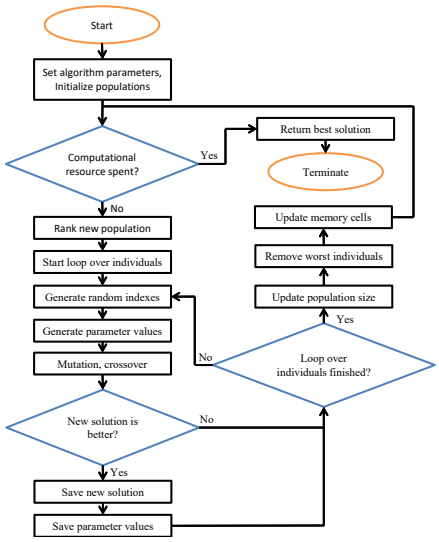

Figure 1. Flow chart of the L-NTADE algorithm.

4. Experimental Setup and Results

4.1. Benchmark Functions and Parameters

The main idea of this study is to propose a different algorithmic scheme for DE, so the experiments with L-NTADE are aimed at evaluating the sensitivity to the newest and top populations' size as well as to the used mutation strategy. The experiments are performed on two benchmark suites, namely the CEC 2017 [35] and 2022 [36] Single Objective Bound Constrained Numerical Optimization problems. These two benchmarks were chosen as they have different settings, in particular, the available number of function evaluations in CEC 2017 is smaller compared to CEC 2022, which makes it possible to evaluate the efficiency of the proposed algorithm in different usage scenarios.

The CEC 2017 benchmark consists of 30 test functions with dimensions 10, 30, 50 and 100, the computational resource is set to $10{,}000D$ function evaluations (1×10^5, 3×10^5, 5×10^5 and 1×10^6 correspondingly), and 51 independent runs are made for every dimension and function.

The CEC 2022 benchmark consists of 12 test functions with dimensions 10 and 20, with computational resource set to 2×10^5 and 1×10^6 evaluations, and 30 independent runs are made for every test function and dimension.

The proposed algorithm was implemented in C++, compiled with GCC, and ran on 8 AMD Ryzen 3700 PRO and 7 AMD Ryzen 1700 with 8 cores each under Ubuntu Linux 20.04. The computations were paralleled using OpenMPI 4.0.3, and the network file system (NFS) was used to store the results. The post-processing of the results, statistical tests and visualizations were performed in Python 3.6.

4.2. Numerical Results

To test the L-NTADE algorithm, the initial population size N_{max}, mutation strategy, selective pressure parameter kp and scaling factor adaptation bias pm were changed. N_{max} changed from $15D$ to $25D$ with step $5D$, four mutation strategies were considered, which resulted in 150 to 250 individuals for $D = 10$ and 1500 to 2500 individuals for $D = 100$. Further increase of the population size parameter resulted in performance deterioration in most cases. The selective pressure was applied to the $r2$ index in all mutation strategies, the ranking procedure was applied to the top population in *r-new-to-ptop/t/t* and *r-new-to-ptop/t/n*, and to the newest population in two other mutations. The selective pressure was applied only to $r2$ as the preliminary tests have shown that applying it to other indexes does not bring any benefits. Two controlling values were used, $kp = 0$ and $kp = 3$, with the former resulting in zero selective pressure (uniform distribution). The biased parameter adaptation was applied only for scaling factor F as previous studies have shown that it has little effect on the Cr. The tested values are $pm = 2$, the same as in L-SHADE, and $pm = 4$, resulting in larger F values.

To compare the efficiency of different variants of L-NTADE, two main instruments were used, the Mann–Whitney rank sum statistical test with normal approximation and tie-breaking to compare a pair of variants. The normal approximation in the Mann–Whitney test means that the resulting statistics is the standard score (Z-score). This simplifies reasoning and allows Z-scores to be used directly to evaluate the level of difference between a pair of algorithms. As the number of experiments for every function and dimension is relatively large, 51 for CEC 2017 and 30 for CEC 2022, the usage of normal approximation is justified. Therefore, in later tables and figures the standard score values and total standard score over all test functions will be used to compare the efficiency of two algorithms along with the number of wins, ties and/or losses. In cases where a conclusion about the significance of the difference is required, the significance level will be set to 0.01. In Table 1, each cell contains the number of wins/ties/losses and the total standard score summed over all test functions when comparing L-NTADE and NL-SHADE-LBC, which took second place in the CEC 2022 competition.

The comparison in Table 1 shows that the proposed L-NTADE algorithm can be better or worse than the NL-SHADE-LBC depending on the used parameters. For example, applying biased parameter adaptation always gives much worse performance, while the selective pressure has a positive effect if the population size and problem dimension is larger. As for the mutation strategies, in the $10D$ case, the *r-new-to-ptop/t/n* have shown the best results, but other strategies have shown similar efficiency with and without selective pressure. In the $20D$ case, the best strategy is *r-new-to-ptop/n/t* combined with selective pressure and increased population size. However, without both modifications, it also performs better than other strategies.

Tables 2 and 3 contain the results of testing L-NTADE on the CEC 2017 benchmark. The competitor chosen for L-NTADE is the L-SHADE-RSP algorithm, the second-best approach in the CEC 2018 competition, which used the same benchmark. The same combinations of parameters were tested as in Table 1.

The results in Tables 2 and 3 show that the *r-new-to-ptop/n/t* strategy combined with selective pressure and biased parameter adaptation has the best performance in most cases. However, in the $10D$ case, the *r-new-to-ptop/t/t* strategy has the best performance with $N_{max} = 20D$, and the difference between variants with smaller and larger population sizes is rather small. Other strategies here have much worse results. Nevertheless, the L-NTADE is mostly superior compared to the L-SHADE-RSP algorithm. In the $30D$ case, the *r-new-to-ptop/n/t* with $N_{max} = 20D$, selective pressure and $pm = 4$ has the best performance, winning L-SHADE-RSP on 17 functions out of 30, and losing in only two cases. The second-best strategy here is *r-new-to-ptop/n/n*, which uses individuals only from the newest population in the second difference. For other mutation strategies, it can be observed that biased parameter adaptation significantly increases the performance of the L-NTADE.

Table 1. L-NTADE vs. NL-SHADE-LBC, CEC 2022, Mann–Whitney tests and total standard score.

$D = 10$					
Selective pressure		$kp = 0$		$kp = 3$	
Scaling factor adaptation bias		$pm = 2$	$pm = 4$	$pm = 2$	$pm = 4$
$N_{max} = 15D$	*r-new-to-ptop/t/t*	6/3/3 19.8	0/5/7 −45.2	6/2/4 20.2	0/6/6 −44.5
	r-new-to-ptop/t/n	**6/4/2** **21.9**	1/4/7 −40.2	5/4/3 19.9	1/4/7 −42.5
	r-new-to-ptop/n/t	6/2/4 18.4	1/2/9 −42.6	6/3/3 17.4	1/3/8 −41.1
	r-new-to-ptop/n/n	6/3/3 19.4	1/3/8 −40.3	6/3/3 20.4	1/4/7 −42.2
$N_{max} = 20D$	*r-new-to-ptop/t/t*	4/5/3 3.9	0/4/8 −48.7	4/5/3 3.7	0/6/6 −40.7
	r-new-to-ptop/t/n	3/6/3 2.7	0/5/7 −44.8	3/5/4 3.8	0/4/8 −44.9
	r-new-to-ptop/n/t	3/3/6 −20.3	0/3/9 −48.0	5/4/3 14.8	1/4/7 −40.2
	r-new-to-ptop/n/n	3/4/5 −13.4	0/5/7 −43.3	5/5/2 15.9	0/5/7 −39.1
$N_{max} = 25D$	*r-new-to-ptop/t/t*	0/4/8 −41.7	0/4/8 −47.9	0/4/8 −39.5	0/5/7 −44.0
	r-new-to-ptop/t/n	0/6/6 −38.1	0/5/7 −44.3	0/5/7 −40.5	0/4/8 −49.3
	r-new-to-ptop/n/t	0/3/9 −51.0	0/4/8 −52.3	2/2/8 −28.3	0/3/9 −50.4
	r-new-to-ptop/n/n	0/6/6 −42.3	0/4/8 −50.2	2/6/4 −25.4	0/4/8 −46.3
$D = 20$					
Selective pressure		$kp = 0$		$kp = 3$	
Scaling factor adaptation bias		$pm = 2$	$pm = 4$	$pm = 2$	$pm = 4$
$N_{max} = 15D$	*r-new-to-ptop/t/t*	5/4/3 8.8	4/4/4 −4.8	5/5/2 13.1	4/4/4 −2.9
	r-new-to-ptop/t/n	5/5/2 18.8	2/5/5 −17.9	6/3/3 18.0	3/4/5 −17.1
	r-new-to-ptop/n/t	6/5/1 25.6	3/4/5 −18.0	5/6/1 21.8	4/4/4 −6.2
	r-new-to-ptop/n/n	5/5/2 19.5	4/3/5 −10.1	6/3/3 20.8	3/4/5 −8.2
$N_{max} = 20D$	*r-new-to-ptop/t/t*	5/4/3 15.0	3/4/5 −16.3	5/4/3 10.4	3/4/5 −14.5
	r-new-to-ptop/t/n	6/3/3 21.0	3/4/5 −15.8	6/3/3 23.0	2/5/5 −18.1
	r-new-to-ptop/n/t	6/4/2 25.0	3/4/5 −13.0	5/5/2 26.7	3/4/5 −13.4
	r-new-to-ptop/n/n	7/2/3 24.4	4/3/5 −10.9	6/3/3 25.5	4/3/5 −7.5
$N_{max} = 25D$	*r-new-to-ptop/t/t*	5/4/3 10.1	3/4/5 −14.1	5/4/3 7.0	3/4/5 −14.6
	r-new-to-ptop/t/n	5/4/3 21.3	3/4/5 −14.3	7/2/3 24.7	3/4/5 −16.0
	r-new-to-ptop/n/t	4/5/3 3.9	3/4/5 −11.9	**6/4/2** **27.5**	3/5/4 −12.1
	r-new-to-ptop/n/n	6/4/2 23.1	4/3/5 −7.5	5/5/2 25.4	4/3/5 −5.4

Table 2. L-NTADE vs. L-SHADE-RSP, CEC 2017, $10D$ **and** $30D$**, Mann–Whitney tests and total standard score.**

$D = 10$					
Selective pressure		$kp = 0$		$kp = 3$	
Scaling factor adaptation bias		$pm = 2$	$pm = 4$	$pm = 2$	$pm = 4$
$N_{max} = 15D$	*r-new-to-ptop/t/t*	8/16/6 8.9	9/16/5 20.5	7/18/5 5.9	9/15/6 16.9
	r-new-to-ptop/t/n	7/16/7 −6.2	8/13/9 −5.0	6/17/7 −7.4	6/15/9 −9.7
	r-new-to-ptop/n/t	5/19/6 −2.2	7/15/8 1.5	4/18/8 −20.2	5/14/11 −24.0
	r-new-to-ptop/n/n	7/18/5 0.7	6/19/5 8.4	7/18/5 −0.9	7/16/7 0.2
$N_{max} = 20D$	*r-new-to-ptop/t/t*	7/18/5 18.4	9/16/5 27.2	9/16/5 20.7	**10/14/6** **28.9**
	r-new-to-ptop/t/n	7/17/6 0.4	8/13/9 4.6	7/16/7 −0.3	8/15/7 9.2
	r-new-to-ptop/n/t	7/17/6 12.2	7/15/8 5.3	8/14/8 −7.3	8/12/10 −12.8
	r-new-to-ptop/n/n	7/19/4 2.6	7/18/5 16.3	5/20/5 5.9	7/16/7 3.7
$N_{max} = 25D$	*r-new-to-ptop/t/t*	8/17/5 20.5	9/16/5 22.1	9/16/5 24.2	9/17/4 25.1
	r-new-to-ptop/t/n	8/17/5 14.3	7/16/7 8.7	7/19/4 14.4	7/14/9 4.7
	r-new-to-ptop/n/t	8/18/4 17.7	8/15/7 11.1	7/17/6 4.6	8/13/9 −11.2
	r-new-to-ptop/n/n	8/20/2 13.4	6/19/5 12.4	8/14/8 6.3	6/20/4 7.1
$D = 30$					
Selective pressure		$kp = 0$		$kp = 3$	
Scaling factor adaptation bias		$pm = 2$	$pm = 4$	$pm = 2$	$pm = 4$
$N_{max} = 15D$	*r-new-to-ptop/t/t*	15/5/10 45.1	17/9/4 118.8	15/5/10 43.5	17/10/3 117.0
	r-new-to-ptop/t/n	14/7/9 33.5	17/10/3 108.2	15/7/8 33.7	17/10/3 109.8
	r-new-to-ptop/n/t	17/6/7 84.4	18/10/2 120.8	18/7/5 103.4	18/10/2 121.2
	r-new-to-ptop/n/n	15/8/7 57.3	19/8/3 117.7	16/8/6 76.7	19/8/3 124.0
$N_{max} = 20D$	*r-new-to-ptop/t/t*	15/5/10 47.5	17/10/3 118.0	15/6/9 45.8	17/10/3 116.5
	r-new-to-ptop/t/n	14/8/8 39.5	17/10/3 104.9	15/6/9 44.0	17/10/3 102.2
	r-new-to-ptop/n/t	17/6/7 85.0	18/9/3 119.9	18/8/4 112.6	**17/11/2** **124.7**
	r-new-to-ptop/n/n	16/7/7 69.1	19/8/3 119.4	16/8/6 87.2	19/8/3 123.2
$N_{max} = 25D$	*r-new-to-ptop/t/t*	15/4/11 49.0	16/11/3 106.3	15/6/9 52.8	17/10/3 116.7
	r-new-to-ptop/t/n	15/6/9 43.7	16/11/3 94.0	16/5/9 43.1	17/10/3 102.1
	r-new-to-ptop/n/t	18/5/7 91.4	16/11/3 111.1	18/8/4 110.6	17/11/2 117.0
	r-new-to-ptop/n/n	17/6/7 74.7	17/10/3 109.2	17/8/5 93.8	17/10/3 114.7

Table 3. L-NTADE vs. L-SHADE-RSP, CEC 2017, $50D$ and $100D$, Mann–Whitney tests and total standard score.

$D = 50$					
Selective pressure		$kp = 0$		$kp = 3$	
Scaling factor adaptation bias		$pm = 2$	$pm = 4$	$pm = 2$	$pm = 4$
$N_{max} = 15D$	*r-new-to-ptop/t/t*	13/3/14 −2.0	17/8/5 94.4	13/2/15 −2.2	16/8/6 94.1
	r-new-to-ptop/t/n	12/1/17 −25.9	16/6/8 68.4	13/0/17 −26.1	16/6/8 67.2
	r-new-to-ptop/n/t	13/6/11 18.6	19/7/4 119.2	15/5/10 48.0	21/7/2 140.3
	r-new-to-ptop/n/n	13/1/16 −12.2	17/7/6 82.7	12/4/14 −0.5	17/7/6 98.2
$N_{max} = 20D$	*r-new-to-ptop/t/t*	13/4/13 3.4	18/8/4 103.1	13/4/13 3.6	18/8/4 98.5
	r-new-to-ptop/t/n	13/2/15 −17.5	16/6/8 69.4	13/2/15 −16.9	16/7/7 71.1
	r-new-to-ptop/n/t	13/7/10 33.5	19/7/4 122.3	15/7/8 54.2	**20/8/2** **141.8**
	r-new-to-ptop/n/n	13/3/14 −8.5	17/7/6 87.4	12/5/13 4.9	18/7/5 105.9
$N_{max} = 25D$	*r-new-to-ptop/t/t*	13/3/14 1.8	17/10/3 96.2	13/4/13 7.3	16/10/4 94.9
	r-new-to-ptop/t/n	13/3/14 −10.5	16/7/7 72.2	13/3/14 −13.4	15/7/8 71.2
	r-new-to-ptop/n/t	13/6/11 33.4	19/7/4 112.6	16/5/9 62.8	21/7/2 141.5
	r-new-to-ptop/n/n	13/2/15 −6.3	18/5/7 87.4	12/7/11 6.0	18/6/6 106.5
$D = 100$					
Selective pressure		$kp = 0$		$kp = 3$	
Scaling factor adaptation bias		$pm = 2$	$pm = 4$	$pm = 2$	$pm = 4$
$N_{max} = 15D$	*r-new-to-ptop/t/t*	11/0/19 −65.8	14/7/9 50.7	11/0/19 −66.2	14/7/9 46.1
	r-new-to-ptop/t/n	11/0/19 −68.7	12/2/16 −2.5	10/1/19 −73.1	12/3/15 1.3
	r-new-to-ptop/n/t	11/1/18 −49.8	16/6/8 77.2	11/3/16 −37.3	19/5/6 107.3
	r-new-to-ptop/n/n	10/1/19 −67.2	14/3/13 26.0	10/1/19 −63.5	15/5/10 54.6
$N_{max} = 20D$	*r-new-to-ptop/t/t*	11/0/19 −64.8	15/6/9 62.4	11/0/19 −63.7	16/5/9 60.8
	r-new-to-ptop/t/n	10/1/19 −69.1	12/5/13 11.4	10/1/19 −69.9	13/4/13 13.4
	r-new-to-ptop/n/t	12/1/17 −40.3	19/3/8 86.6	12/1/17 −31.6	20/4/6 120.5
	r-new-to-ptop/n/n	10/1/19 −68.2	14/6/10 32.8	10/1/19 −63.1	15/7/8 65.1
$N_{max} = 25D$	*r-new-to-ptop/t/t*	11/0/19 −64.5	17/4/9 63.1	11/0/19 −65.5	17/4/9 64.5
	r-new-to-ptop/t/n	10/1/19 −70.5	13/5/12 19.4	10/1/19 −67.1	12/5/13 14.1
	r-new-to-ptop/n/t	12/1/17 −43.3	18/4/8 88.7	12/0/18 −32.8	**22/3/5** **126.3**
	r-new-to-ptop/n/n	10/1/19 −66.7	14/6/10 32.4	10/0/20 −66.1	17/5/8 76.8

In the $50D$ case, the best-performing variant is exactly the same, i.e., *r-new-to-ptop/n/t* with $N_{max} = 20D$, selective pressure and $pm = 4$. Here, it can be observed that the effect of the population size is minor, unlike experiments on the CEC 2022 benchmark. The

exponential rank-based selective pressure improves the performance in most cases, but this improvement is rather limited, unlike the improvement due to the biased parameter adaptation. In the $100D$ case, the same conclusions are applicable, although the selective pressure here has a much larger effect, especially for the *r-new-to-ptop/n/t* mutation strategy.

Considering the results above, for the next experiments, the following parameters were chosen. For CEC 2022, $N_{max} = 15D$, $kp = 0$, $pm = 2$, and the mutation strategy is *r-new-to-ptop/n/t*. For CEC 2017, the population size is changed to $N_{max} = 20D$, $kp = 3$, $pm = 4$, and the mutation strategy is *r-new-to-ptop/n/t*. Table 4 shows the comparison of L-NTADE with alternative approaches on the CEC 2022 benchmark, including the top three algorithms (EA4eig [54], NL-SHADE-LBC [55] and NL-SHADE-RSP-MID [56]). The values in the table are the number of wins/ties/losses (total standard score).

Table 4. Mann–Whitney tests of L-NTADE against the competition top 3, CEC 2022, and other approaches, number of wins/ties/losses and total standard score.

Algorithm	**10*D***	**20*D***
L-NTADE	0/12/0 (0.0)	0/12/0 (0.0)
EA4eig [54]	6/2/4 (6.94)	6/2/4 (9.38)
NL-SHADE-LBC [55]	6/2/4 (18.40)	6/5/1 (25.63)
NL-SHADE-RSP-MID [56]	5/3/4 (8.69)	8/1/3 (36.29)
L-SHADE-RSP [49]	7/1/4 (25.19)	5/5/2 (17.71)
NL-SHADE-RSP [51]	7/2/3 (26.78)	8/3/1 (39.26)
MLS-LSHADE [52]	8/1/3 (31.93)	6/2/4 (20.11)
APGSK-IMODE [57]	7/3/2 (38.99)	9/1/2 (47.44)
MadDE [58]	9/2/1 (46.54)	8/2/2 (37.17)

As Table 4 shows, L-NTADE is capable of outperforming the best algorithms participating in the CEC 2022 competition as well as other approaches. The summed standard score gives additional information, allowing different performance levels to be observed with similar numbers of wins and losses. Table 5 shows the comparison on the CEC 2017 benchmark, and the same notation is used.

Table 5. Mann–Whitney tests of L-NTADE against other approaches, CEC 2017, number of wins/ties/losses and total standard score.

Algorithm	**10*D***	**30*D***
L-NTADE	0/30/0 (0.0)	0/30/0 (0.0)
L-SHADE-RSP [49]	8/12/10 (-12.81)	17/11/2 (124.67)
LSHADE-SPACMA [50]	9/12/9 (-17.63)	13/10/7 (50.41)
jSO [48]	7/13/10 (-14.98)	18/11/1 (133.69)
EBOwithCMAR [59]	6/13/11 (-46.46)	15/9/6 (66.78)
Algorithm	**50*D***	**100*D***
L-NTADE	0/30/0 (0.0)	0/30/0 (0.0)
L-SHADE-RSP [49]	20/8/2 (141.76)	20/4/6 (120.46)
LSHADE-SPACMA [50]	13/5/12 (2.26)	11/3/16 (-33.34)
jSO [48]	21/7/2 (158.53)	24/2/4 (147.53)
EBOwithCMAR [59]	19/6/5 (114.56)	19/4/7 (94.46)

The performance of L-NTADE in the $10D$ case, according to Table 5, is inferior compared to other approaches, while for other dimensions it is much better. This can be explained by the fact that the best mutation strategy for $10D$ is not the one used in the experiments in Table 5. This is done for versatility. In the $30D$ case, L-NTADE outperforms all other methods, but in $50D$ it has similar performance to LSHADE-SPACMA. In $100D$, the only algorithm with better performance is again LSHADE-SPACMA, probably due to the hybridization with CMA-ES.

To evaluate the computational efficiency of L-NTADE, the CEC 2022 benchmark was used, for which the time complexity is estimated by calculating the time required to calculate a set of mathematical expressions ($T0$), evaluate the first test function ($T1$)

and run the algorithm on this test function five times to obtain the average ($T2$). The resulting value is calculated as $(T2 - T1)/T0$ [36]. The comparison results of L-NTADE with NL-SHADE-RSP and NL-SHADE-LBC are given in Table 6.

Table 6. Computational complexity of L-NTADE compared with NL-SHADE-LBC and NL-SHADE-RSP on CEC 2022 benchmark.

L-NTADE				
D	$T0$	$T1$	$T2$	$(T2 - T1)/T0$
$D = 10$	8×10^{-6}	2.4×10^{-5}	1.268×10^{-4}	12.85
$D = 20$	8×10^{-6}	7.2×10^{-5}	2.052×10^{-4}	16.65
NL-SHADE-LBC				
D	$T0$	$T1$	$T2$	$(T2 - T1)/T0$
$D = 10$	8×10^{-6}	2.4×10^{-5}	1.330×10^{-4}	13.63
$D = 20$	8×10^{-6}	7.2×10^{-5}	2.570×10^{-4}	23.15
NL-SHADE-RSP				
D	$T0$	$T1$	$T2$	$(T2 - T1)/T0$
$D = 10$	8×10^{-6}	2.3×10^{-5}	1.110×10^{-4}	11.00
$D = 20$	8×10^{-6}	7.3×10^{-5}	1.774×10^{-4}	13.05

The comparison in Table 6 demonstrates that the complexity of L-NTADE is comparable with other similar approaches, and the usage of an additional population does not result in significant additional effort.

The presented results of the experiments have shown that the algorithmic scheme of L-NTADE has a certain potential, but for a better understanding of the reasons of its performance, several additional tests were performed. For a deeper dive into the algorithm, the histograms of pairwise Euclidean distances between all individuals were built on every generation for both the top and the newest population. These histograms were color-coded and built on a heatmap together with the average distance. Figure 2 show these histograms on F5, CEC 2017, shifted and rotated Rastrigin's Function, 10D, Figure 3 shows F17, hybrid Function (7) (Katsuura, Ackley's, Expanded Griewank's plus Rosenbrock's, Modified Schwefel's, Rastrigin's) and Figure 4 shows F29, composition Function (9) (hybrid Function (5), hybrid Function (8), hybrid Function (9)) [35].

The distance histograms in Figure 2 demonstrate a significant difference between the top and newest populations. In particular, the top population after the initial convergence process remains in a relatively steady state, as can be seen by many horizontal lines, each corresponding to a point near a local minimum (which are intrinsic for Rastrigin's function). The newest population, on the other hand, is continuously updated, resulting in a noise-like image. After the first third, when populations are relatively similar, they split their roles into keeping the information about potentially interesting solutions and actively searching for better ones. At the end of the search, it can be observed that the top population stays with four best solutions, while the newest continues attempts to improve.

Figure 3 shows a similar situation, where both populations tend to converge in the first 500 generations and then split into many groups corresponding to local optima. At around generation 900, one of the areas of local search dominates the other ones, which are deleted, and another phase of active convergence begins. At a certain point after generation 1300, the top population is stuck, which is again seen by horizontal lines, but the newest population continues the search, giving a similar noise-like picture in the second half of the search process.

In Figure 4, similar trends can be observed. However, now there are stages when even the newest population may get stuck, for example around generation 1700. Nevertheless, the search process continues further, generating different solutions in the newest population and transferring them into the top population.

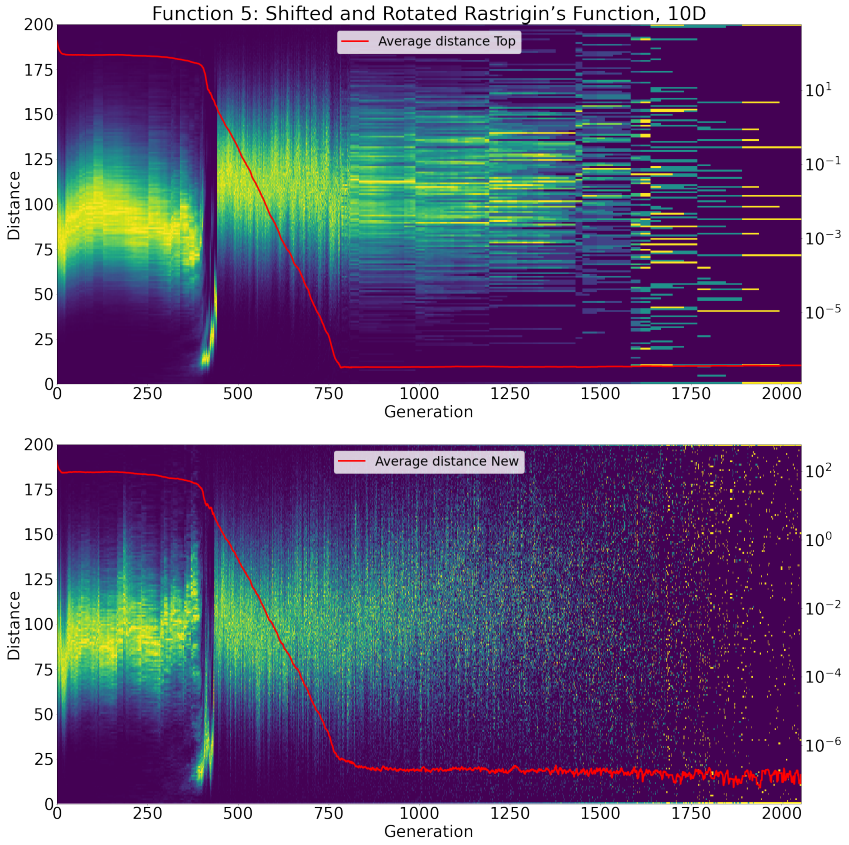

Figure 2. Heatmap of pairwise distance histograms on every generation, F5, CEC 2017, 10*D*.

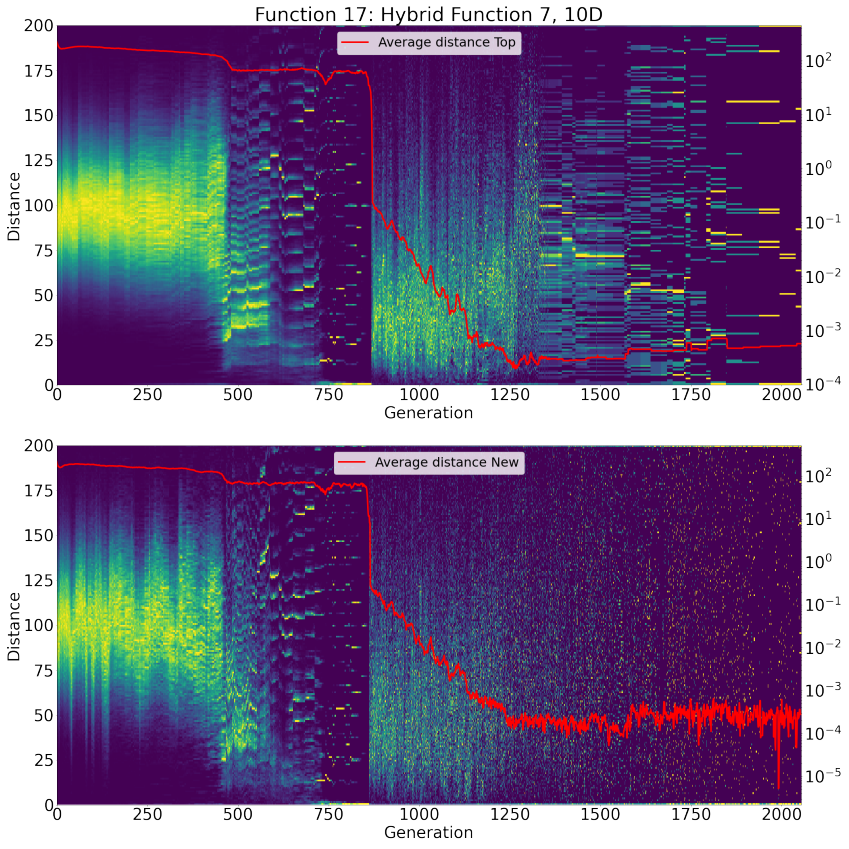

Figure 3. Heatmap of pairwise distance histograms on every generation, F17, CEC 2017, 10*D*.

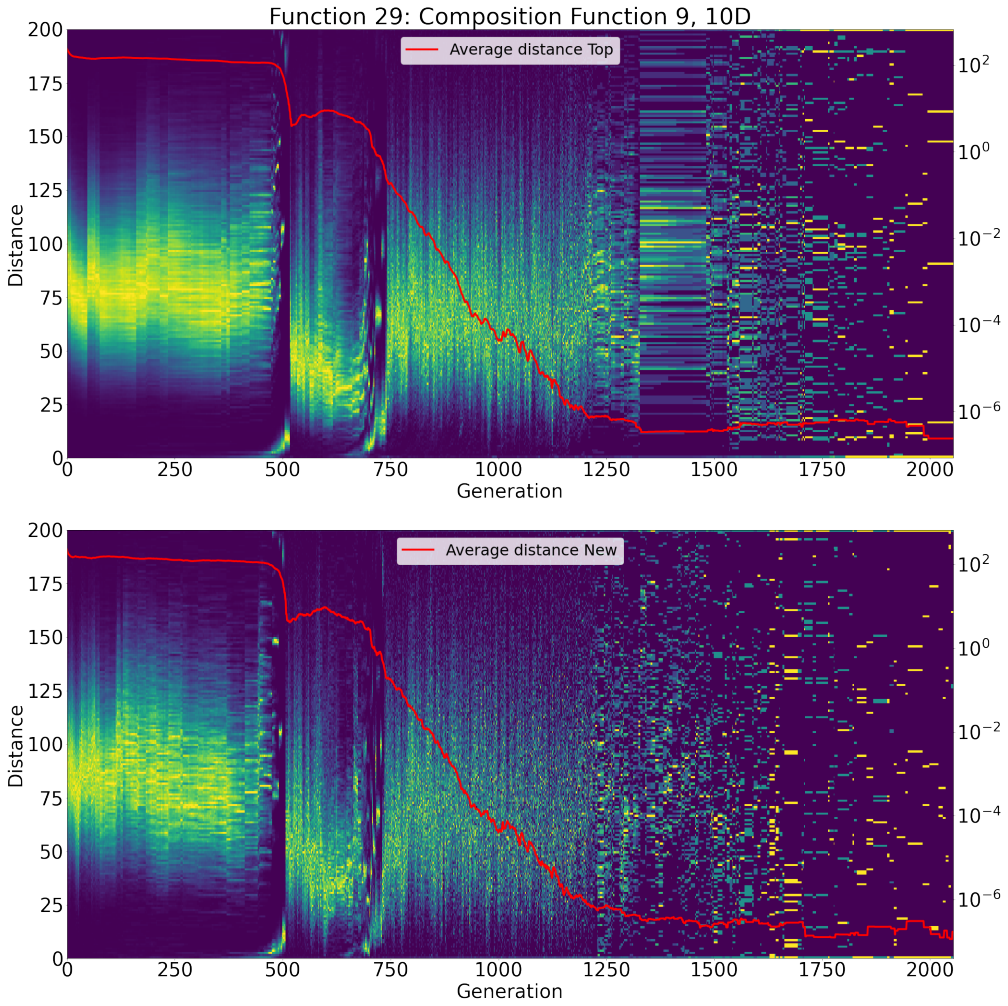

Figure 4. Heatmap of pairwise distance histograms on every generation, F29, CEC 2017, 10*D*.

5. Discussion

The experimental results in the previous section have demonstrated that the concepts proposed in the UDE algorithm [34] can be efficiently utilized, for example, in the way it is done in L-NTADE. The proposed approach uses a specific update rule for the newest population, which constantly replaces solutions with more efficient ones, but unlike classical DE selection, there is a chance that a better solution will be replaced by a worse one. At the same time, all solutions with high fitness are always stored in the top population. The ongoing update of the newest population is probably one of the reasons for the different behavior of L-NTADE compared to NL-SHADE-LBC, L-SHADE-RSP and other methods, which was observed in distance histograms. Additionally, utilizing two populations instead of one allows L-NTADE to solve some of the problems more efficiently, especially relatively complex hybrid and composition functions.

As for the mutation strategies tested, *r-new-to-ptop/n/t*, *r-new-to-ptop/t/n* and *r-new-to-ptop/n/n* are of interest. Combining the top and new vectors in the second difference appears to have a positive effect on the final efficiency. Additionally, the *r-new-to-ptop/n/t* strategy, which performed best overall, uses a directed second difference. The first difference between the randomly chosen newest and one of the pb% top vectors is in fact a point on a line connecting these two vectors, and the position of this point is controlled by F. The second difference does a similar thing, i.e., it makes a step from the newest vector to one of the top vectors, i.e., towards better solutions. In the experiments with rank-based selective pressure in L-SHADE-RSP, NL-SHADE-RSP and NL-SHADE-LBC, a similar structure of

mutation strategy was used, i.e., in the second difference, the direction of step is mainly towards better solutions. This statement can also be supported by the fact that *r-new-to-ptop/n/t* was able to perform better than most other methods in the $100D$ case of CEC 2017 functions, and in a study on selective pressure effects [42], it was shown that larger selective pressure has a positive effect in high-dimensional cases. At the same time, going in the opposite direction, from better solutions to randomly chosen ones when using *r-new-to-ptop/t/n* does not bring any benefits. Of course, other mutation strategies can be proposed for L-NTADE, but testing all possible variants is beyond the scope of the current study.

One of the disadvantages of L-NTADE is that it can be very sensitive to the population size, as the experiments on CEC 2022 have shown. However, for CEC 2017 this was not the case. Moreover, for CEC 2017, the selective pressure and biased scaling factor F adaptation worked well, but failed for CEC 2022. Considering the fact that these benchmarks mainly differ in the amount of computational resources available, a conclusion can be drawn. If the computational resource is relatively small, around $10000D$, then selective pressure and biased parameter adaptation should be used. Otherwise, the population size should be chosen carefully, but selective pressure may help to achieve better results with a large population size. The tested version of L-NTADE is a baseline, and it can be further improved by introducing modifications proposed for other DE-based approaches. For example:

1. Adding an archive set and a specifically developed update strategies for it;
2. Adding crossover rate sorting;
3. Introducing a control strategy for the pb% parameter;
4. Developing new parameter adaptation strategies, suitable for L-NTADE;
5. Developing adaptive mechanisms for switching between mutation strategies during the algorithm run;
6. Creating hybrids of L-NTADE with other approaches.

The mentioned possible ways of improving L-NTADE are subjects for further studies.

6. Conclusions

This study proposed a new algorithmic scheme for differential evolution, which uses two populations and new mutation strategies. The performed experiments have shown that the developed L-NTADE is a highly competitive approach, which is able to outperform some of the state-of-the-art algorithms on popular benchmarks CEC 2017 and CEC 2022, especially on complex multi-modal functions. The proposed algorithm is relatively easy to implement as it is a non-hybrid method, and it can be further improved by adding modifications proposed for other DE methods. Unlike most of the DE versions, L-NTADE does not use the greedy selection strategy, but instead maintains two populations, one keeping the best solutions, with the other continuously updating. The results and analysis of algorithm behavior have demonstrated the advantages of such a scheme, i.e., the algorithm keeps the search process running all the time. One of the drawbacks of L-NTADE is its sensitivity to the population size parameter. However, all known DE algorithms have the same problem. Further studies of the proposed algorithmic scheme may include experimenting with replacement strategies in the population of new individuals and setting a different size for the new and top populations with specific control strategies.

Author Contributions: Conceptualization, V.S. and S.A.; methodology, V.S., S.A. and E.S.; software, V.S. and E.S.; validation, V.S., S.A. and E.S.; formal analysis, S.A.; investigation, V.S.; resources, E.S. and V.S.; data curation, E.S.; writing—original draft preparation, V.S. and S.A.; writing—review and editing, V.S.; visualization, S.A. and V.S.; supervision, E.S.; project administration, E.S. funding acquisition, S.A. and V.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Ministry of Science and Higher Education of the Russian Federation, Grant No. 075-15-2022-1121.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

CI	Computational Intelligence
NN	Neural Networks
FL	Fuzzy Logic
SI	Swarm Intelligence
EA	Evolutionary Algorithms
GA	Genetic Algorithms
ES	Evolutionary Strategies
PSO	Particle Swarm Optimization
DE	Differential Evolution
CEC	Congress on Evolutionary Computation
SHADE	Success-History Adaptive Differential Evolution
LPSR	Linear Population Size Reduction
NLPSR	Non-Linear Population Size Reduction
LBC	Linear Bias Change
UDE	Unbounded Differential Evolution
L-NTADE	Linear population size reduction Newest and Top Adaptive Differential Evolution

References

1. Sloss, A.N.; Gustafson, S. 2019 Evolutionary Algorithms Review. In Proceedings of the Genetic Programming Theory and Practice, East Lansing, MI, USA, 16–19 May 2019.
2. Sinha, A.; Malo, P.; Deb, K. A Review on Bilevel Optimization: From Classical to Evolutionary Approaches and Applications. *IEEE Trans. Evol. Comput.* **2018**, *22*, 276–295. [CrossRef]
3. Alkayem, N.F.; Cao, M.; Shen, L.; Fu, R.; Sumarac, D. The combined social engineering particle swarm optimization for real-world engineering problems: A case study of model-based structural health monitoring. *Appl. Soft Comput.* **2022**, *123*, 108919. [CrossRef]
4. Alkayem, N.F.; Shen, L.; Al-hababi, T.; Qian, X.; Cao, M. Inverse Analysis of Structural Damage Based on the Modal Kinetic and Strain Energies with the Novel Oppositional Unified Particle Swarm Gradient-Based Optimizer. *Appl. Sci.* **2022**, *12*, 11689. [CrossRef]
5. Price, K.; Storn, R.; Lampinen, J. *Differential Evolution: A Practical Approach to Global Optimization*; Springer: Berlin/Heidelberg, Germany, 2005.
6. Ali, M.; Awad, N.H.; Suganthan, P.; Shatnawi, A.; Reynolds, R. An improved class of real-coded Genetic Algorithms for numerical optimization. *Neurocomputing* **2018**, *275*, 155–166. [CrossRef]
7. Maheswaranathan, N.; Metz, L.; Tucker, G.; Sohl-Dickstein, J. Guided Evolutionary Strategies: Escaping the Curse of Dimensionality in Random Search. 2018. Available online: https://openreview.net/forum?id=B1xFxh0cKX (accessed on 5 November 2022).
8. Bonyadi, M.; Michalewicz, Z. Particle Swarm Optimization for Single Objective Continuous Space Problems: A Review. *Evol. Comput.* **2017**, *25*, 1–54. [CrossRef] [PubMed]
9. Beyer, H.; Sendhoff, B. Simplify your covariance matrix adaptation evolution strategy. *IEEE Trans. Evol. Comput.* **2017**, *21*, 746–759. [CrossRef]
10. Kar, A. Bio inspired computing—A review of algorithms and scope of applications. *Expert Syst. Appl.* **2016**, *59*, 20–32. [CrossRef]
11. Skvorc, U.; Eftimov, T.; Korosec, P. CEC Real-Parameter Optimization Competitions: Progress from 2013 to 2018. In Proceedings of the 2019 IEEE Congress on Evolutionary Computation (CEC), Wellington, New Zealand, 10–13 June 2019; pp. 3126–3133.
12. Das, S.; Suganthan, P. Differential evolution: A survey of the state-of-the-art. *IEEE Trans. Evol. Comput.* **2011**, *15*, 4–31. [CrossRef]
13. Das, S.; Mullick, S.; Suganthan, P. Recent advances in differential evolution—An updated survey. *Swarm Evol. Comput.* **2016**, *27*, 1–30. [CrossRef]
14. Qin, A.; Suganthan, P. Self-adaptive differential evolution algorithm for numerical optimization. In Proceedings of the IEEE Congress on Evolutionary Computation, Edinburgh, UK, 2–5 September 2005; pp. 1785–1791. [CrossRef]
15. dos Santos Coelho, L.; Ayala, H.V.H.; Mariani, V.C. A self-adaptive chaotic differential evolution algorithm using gamma distribution for unconstrained global optimization. *Appl. Math. Comput.* **2014**, *234*, 452–459. [CrossRef]
16. Mallipeddi, R.; Suganthan, P.N.; Pan, Q.; Tasgetiren, M.F. Differential evolution algorithm with ensemble of parameters and mutation strategies. *Appl. Soft Comput.* **2011**, *11*, 1679–1696. [CrossRef]
17. Gong, W.; Fialho, Á.; Cai, Z.; Li, H. Adaptive strategy selection in differential evolution for numerical optimization: An empirical study. *Inf. Sci.* **2011**, *181*, 5364–5386. [CrossRef]
18. Brest, J.; Greiner, S.; Boškovic, B.; Mernik, M.; Žumer, V. Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *IEEE Trans. Evol. Comput.* **2006**, *10*, 646–657. [CrossRef]
19. Zhang, J.; Sanderson, A.C. JADE: Self-adaptive differential evolution with fast and reliable convergence performance. In Proceedings of the 2007 IEEE Congress on Evolutionary Computation, Singapore, 25–28 September 2007; pp. 2251–2258.

20. Tanabe, R.; Fukunaga, A. Success-history based parameter adaptation for differential evolution. In Proceedings of the IEEE Congress on Evolutionary Computation, Cancun, Mexico, 20–23 June 2013; IEEE Press: Piscataway, NJ, USA, 2013; pp. 71–78. [CrossRef]
21. Tanabe, R.; Fukunaga, A. Improving the search performance of SHADE using linear population size reduction. In Proceedings of the IEEE Congress on Evolutionary Computation, CEC, Beijing, China, 6–11 July 2014; pp. 1658–1665. [CrossRef]
22. Piotrowski, A.P.; Napiorkowski, J.J. Step-by-step improvement of JADE and SHADE-based algorithms: Success or failure? *Swarm Evol. Comput.* **2018**, *43*, 88–108. [CrossRef]
23. Sun, G.; Xu, G.; Jiang, N. A simple differential evolution with time-varying strategy for continuous optimization. *Soft Comput.* **2020**, *24*, 2727–2747. [CrossRef]
24. Sun, G.; Yang, B.; Yang, Z.; Xu, G. An adaptive differential evolution with combined strategy for global numerical optimization. *Soft Comput.* **2020**, *24*, 6277–6296. [CrossRef]
25. Huynh, T.N.; Do, D.T.T.; Lee, J. Q-Learning-based parameter control in differential evolution for structural optimization. *Appl. Soft Comput.* **2021**, *107*, 107464. [CrossRef]
26. Song, Y.; Wu, D.; Deng, W.; Zhi Gao, X.; Li, T.; Zhang, B.; Li, Y. MPPCEDE: Multi-population parallel co-evolutionary differential evolution for parameter optimization. *Energy Convers. Manag.* **2021**, *228*, 113661. [CrossRef]
27. Tan, Z.; Tang, Y.; Li, K.; Huang, H.; Luo, S. Differential evolution with hybrid parameters and mutation strategies based on reinforcement learning. *Swarm Evol. Comput.* **2022**, *75*, 101194. [CrossRef]
28. Stanovov, V.; Akhmedova, S.; Semenkin, E. The automatic design of parameter adaptation techniques for differential evolution with genetic programming. *Knowl. Based Syst.* **2022**, *239*, 108070. [CrossRef]
29. Stanovov, V.; Akhmedova, S.; Semenkin, E. Neuroevolution for parameter adaptation in differential evolution. *Algorithms* **2022**, *15*, 122. [CrossRef]
30. Meng, Z.; Pan, J.S. HARD-DE: Hierarchical ARchive Based Mutation Strategy With Depth Information of Evolution for the Enhancement of Differential Evolution on Numerical Optimization. *IEEE Access* **2019**, *7*, 12832–12854. [CrossRef]
31. Brest, J.; Maucec, M.S.; Boškovic, B. Self-adaptive Differential Evolution Algorithm with Population Size Reduction for Single Objective Bound-Constrained Optimization: Algorithm j21. In Proceedings of the 2021 IEEE Congress on Evolutionary Computation (CEC), Krakow, Poland, 28 June–1 July 2021; pp. 817–824.
32. Mohamed, A.; Hadi, A.A.; Mohamed, A.K.; Awad, N.H. Evaluating the Performance of Adaptive GainingSharing Knowledge Based Algorithm on CEC 2020 Benchmark Problems. In Proceedings of the 2020 IEEE Congress on Evolutionary Computation (CEC), Glasgow, UK, 19–24 July 2020; pp. 1–8.
33. Zhu, Z.; Chen, L.; Yuan, C.; Xia, C. Global replacement-based differential evolution with neighbor-based memory for dynamic optimization. *Appl. Intell.* **2018**, *48*, 3280–3294. [CrossRef]
34. Kitamura, T.; Fukunaga, A. Differential Evolution with an Unbounded Population. In Proceedings of the 2022 IEEE Congress on Evolutionary Computation (CEC), Padua, Italy, 18–23 July 2022.
35. Awad, N.; Ali, M.; Liang, J.; Qu, B.; Suganthan, P. *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.
36. Kumar, A.; Price, K.; Mohamed, A.; Hadi, A.; Suganthan, P.N. *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, 2021.
37. Storn, R.; Price, K. Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [CrossRef]
38. Kitamura, T.; Fukunaga, A. Duplicate Individuals in Differential Evolution. In Proceedings of the 2022 IEEE Congress on Evolutionary Computation (CEC), Padua, Italy, 18–23 July 2022.
39. Kumar, A.; Biswas, P.P.; Suganthan, P.N. Differential evolution with orthogonal array-based initialization and a novel selection strategy. *Swarm Evol. Comput.* **2022**, *68*, 101010. [CrossRef]
40. Al-Dabbagh, R.D.; Neri, F.; Idris, N.; Baba, M.S.B. Algorithmic design issues in adaptive differential evolution schemes: Review and taxonomy. *Swarm Evol. Comput.* **2018**, *43*, 284–311. [CrossRef]
41. Biedrzycki, R.; Arabas, J.; Jagodziński, D. Bound constraints handling in Differential Evolution: An experimental study. *Swarm Evol. Comput.* **2019**, *50*, 100453. [CrossRef]
42. Stanovov, V.; Akhmedova, S.; Semenkin, E. Selective Pressure Strategy in differential evolution: Exploitation improvement in solving global optimization problems. *Swarm Evol. Comput.* **2019**, *50*, 100463. [CrossRef]
43. Stanovov, V.; Akhmedova, S.; Semenkin, E. Biased Parameter Adaptation in Differential Evolution. *Inf. Sci.* **2021**, *566*, 215–238. [CrossRef]
44. Zhang, J.; Sanderson, A.C. JADE: Adaptive differential evolution with optional external archive. *IEEE Trans. Evol. Comput.* **2009**, *13*, 945–958. [CrossRef]
45. Stanovov, V.; Akhmedova, S.; Semenkin, E. Archive update strategy influences differential evolution performance. *Adv. Swarm Intell.* **2020**, *12145*, 397–404.
46. Bullen, P. *Handbook of Means and Their Inequalities*; Springer: Dordrecht, The Netherlands, 2003. [CrossRef]

47. Biswas, P.P.; Suganthan, P.N. Large Initial Population and Neighborhood Search incorporated in LSHADE to solve CEC2020 Benchmark Problems. In Proceedings of the 2020 IEEE Congress on Evolutionary Computation (CEC), Glasgow, UK, 19–24 July 2020; pp. 1–7. [CrossRef]
48. Brest, J.; Maučec, M.; Boškovic, B. Single objective real-parameter optimization algorithm jSO. In Proceedings of the IEEE Congress on Evolutionary Computation, San Sebastian, Spain, 5–8 June 2017; IEEE Press: Piscataway, NJ, USA, 2017; pp. 1311–1318. [CrossRef]
49. Stanovov, V.; Akhmedova, S.; Semenkin, E. LSHADE Algorithm with Rank-Based Selective Pressure Strategy for Solving CEC 2017 Benchmark Problems. In Proceedings of the 2018 IEEE Congress on Evolutionary Computation (CEC), Rio de Janeiro, Brazil, 8–13 July 2018; pp. 1–8.
50. Mohamed, A.; Hadi, A.A.; Fattouh, A.; Jambi, K. LSHADE with semi-parameter adaptation hybrid with CMA-ES for solving CEC 2017 benchmark problems. In Proceedings of the 2017 IEEE Congress on Evolutionary Computation (CEC), Donostia-San Sebastián, Spain, 5–8 June 2017; pp. 145–152.
51. Stanovov, V.; Akhmedova, S.; Semenkin, E. NL-SHADE-RSP Algorithm with Adaptive Archive and Selective Pressure for CEC 2021 Numerical Optimization. In Proceedings of the 2021 IEEE Congress on Evolutionary Computation (CEC), Krakow, Poland, 28 June–1 July 2021; pp. 809–816. [CrossRef]
52. Cuong, L.V.; Bao, N.N.; Binh, H.T.T. *Technical Report: A Multi-Start Local Search Algorithm with L-SHADE for Single Objective Bound Constrained Optimization*; Technical Report; SoICT, Hanoi University of Science and Technology: Hanoi, Vietnam, 2021.
53. Viktorin, A.; Senkerik, R.; Pluhacek, M.; Kadavy, T.; Zamuda, A. Distance based parameter adaptation for Success-History based Differential Evolution. *Swarm Evol. Comput.* **2019**, *50*, 100462. [CrossRef]
54. Bujok, P.; Kolenovsky, P. Eigen Crossover in Cooperative Model of Evolutionary Algorithms Applied to CEC 2022 Single Objective Numerical Optimisation. In Proceedings of the 2022 IEEE Congress on Evolutionary Computation (CEC), Padua, Italy, 18–23 July 2022.
55. Stanovov, V.; Akhmedova, S.; Semenkin, E. NL-SHADE-LBC algorithm with linear parameter adaptation bias change for CEC 2022 Numerical Optimization. In Proceedings of the 2022 IEEE Congress on Evolutionary Computation (CEC), Padua, Italy, 18–23 July 2022.
56. Biedrzycki, R.; Arabas, J.; Warchulski, E. A Version of NL-SHADE-RSP Algorithm with Midpoint for CEC 2022 Single Objective Bound Constrained Problems. In Proceedings of the 2022 IEEE Congress on Evolutionary Computation (CEC), Padua, Italy, 18–23 July 2022.
57. Mohamed, A.W.; Hadi, A.A.; Agrawal, P.; Sallam, K.M.; Mohamed, A.K. Gaining-Sharing Knowledge Based Algorithm with Adaptive Parameters Hybrid with IMODE Algorithm for Solving CEC 2021 Benchmark Problems. In Proceedings of the 2021 IEEE Congress on Evolutionary Computation (CEC), Krakow, Poland, 28 June–1 July 2021; pp. 841–848.
58. Biswas, S.; Saha, D.; De, S.; Cobb, A.D.; Das, S.; Jalaian, B. Improving Differential Evolution through Bayesian Hyperparameter Optimization. In Proceedings of the 2021 IEEE Congress on Evolutionary Computation (CEC), Krakow, Poland, 28 June–1 July 2021; pp. 832–840.
59. Kumar, A.; Misra, R.K.; Singh, D. Improving the local search capability of Effective Butterfly Optimizer using Covariance Matrix Adapted Retreat Phase. In Proceedings of the 2017 IEEE Congress on Evolutionary Computation (CEC), Donostia-San Sebastián, Spain, 5–8 June 2017; pp. 1835–1842.