Article

Not peer-reviewed version

Comparative Study of Modern Differential Evolution Algorithms: Perspectives on Mechanisms and Performance

Janez Brest * and Mirjam Sepesy Maučec

Posted Date: 6 May 2025

doi: 10.20944/preprints202503.1551.v2

Keywords: global optimization; differential evolution; benchmark suite; mechanisms; statistical tests; performance

Article

Comparative Study of Modern Differential Evolution Algorithms: Perspectives on Mechanisms and Performance

Janez Brest *, **Mirjam Sepesy Maučec**

Faculty of Electrical Engineering and Computer Science, University of Maribor, SI-2000 Maribor, Slovenia; mirjam.sepesy@um.si
* Correspondence: janez.brest@um.si

Abstract: Since the discovery of the Differential Evolution algorithm, new and improved versions have continuously emerged. In this paper, we review selected algorithms based on differential evolution that have been proposed in recent years. We examine the mechanisms integrated into them and compare the performance of algorithms. To compare their performances, statistical comparisons were used as they enable us to draw reliable conclusions about the algorithms' performances. We use the Wilcoxon signed-rank test for pairwise comparisons and the Friedman test for multiple comparisons. Subsequently, the Mann-Whitney U-score test was added. We conducted not only a cumulative analysis of algorithms, but we also focused on their performances regarding the function family (i.e., unimodal, multimodal, hybrid, and composition functions). Experimental results of algorithms were obtained on problems defined for the CEC'24 Special Session and Competition on Single Objective Real Parameter Numerical Optimization. Problem dimensions of 10, 30, 50, and 100 were analyzed. In this paper, we highlight promising mechanisms for further development and improvements based on the study of the selected algorithms.

Keywords: global optimization; differential evolution; benchmark suite; mechanisms; statistical tests; performance

MSC: 68W50

1. Introduction

Differential Evolution (DE) is a simple and effective evolutionary algorithm for solving global optimization problems in continuous space. It has also been adapted for use in discrete, most commonly binary, spaces. The original version was defined for single-objective optimization and was later extended to multi-objective optimization. This line of research, which goes in many directions, highlights the significance of this metaheuristic.

In this paper, we focus our research on DE for single-objective optimization in continuous space. It was introduced in 1996 by Storn and Price. Since then, numerous modifications and improvements have been proposed. This trend continues today and has been especially noticeable in recent years. In the variety of algorithm improvements, it is difficult to determine which one represents a truly significant enhancement. In the paper, we examine the improvements of the DE algorithm that have been published primarily in the past year of algorithm development.

Each year at the Congress on Evolutionary Computation (CEC), a competition for single-objective real parameter numerical optimization is held, in which DE-based algorithms hold a prominent place. In the year 2024, among the six competing algorithms, four were derived from DE. These four are of particular interest to us in this paper. Three algorithms from the recent past are also included in comparison to assess whether 2024 brings significant progress in development.

In literature, algorithms are compared using different statistical techniques to perform single-problem and multiple-problem analysis. Non-parametric statistical tests are commonly used as they are less restrictive than parametric tests. In this paper, we adapted the same tests and add the Mann-Whitney U-score test, which has been used to determine the winners in the most recent CEC 2024 competition. The computation of the U-score in our research slightly differs from that used in CEC 2024, as will be described later in the paper. The paper is organized as follows. Section 2 provides the necessary background knowledge to understand the main ideas of the paper. This section outlines the basic DE algorithm and briefly describes the statistical tests used in the research. Related work is discussed in Section 3. The algorithms used for comparison are described in Section 4, where their main components are highlighted. The algorithms are complex, with several integrated mechanisms and applied settings. Complete descriptions of them are provided with references to the original papers in which they were defined. Section 5 outlines the methodology for the comparative analysis. Extensive experiments conducted to evaluate the performance of the comparative algorithms are presented in Section 6. This section also discusses the obtained results. Section 7 highlights the most promising mechanisms integrated into algorithms under research. Finally, Section 8 summarizes the findings and concludes the paper, offering suggestions for future research directions.

Contributions of the Paper

This paper makes the following contribution:
It describes the recent advances in DE proposed in 2024, focusing on modifications to enhance the algorithm's effectiveness and efficiency. The algorithms are evaluated across dimensions of 10D, 30D, 50D, and 100D. The improvements are statistically validated using the Wilcoxon signed-rank test, the Friedman test, and the Mann-Whitney U-score test.

2. Preliminary

2.1. Differential Evolution

In this subsection, we review the DE algorithm [1,2], as it serves as the core algorithm for the further advancements studied in this paper. An optimization algorithm aims to identify a vector $\mathbf{x}$ so as to optimize $f(\mathbf{x})$; $\mathbf{x} = (x_1, x_2, \ldots, x_D)$. D is the dimensionality of the function f. The variables' domains are defined by their lower and upper bounds: $x_{j,low}, x_{j,upp}$; $j \in \{1, \ldots, D\}$.

DE is a population-based algorithm where each individual in the population is represented by a vector $\mathbf{x}_{i,g}$; $i = 1, 2, \ldots, Np$. Np denotes the population size, and g represents the generation number. During each generation, DE applies mutation, crossover, and selection operations to each vector (target vector) to generate a trial vector (offspring).

Mutation generates the mutant vector $\mathbf{v}_{i,g+1}$ according to:

$$\mathbf{v}_{i,g+1} = \mathbf{x}_{r_1,g} + F \cdot (\mathbf{x}_{r_2,g} - \mathbf{x}_{r_3,g}). \tag{1}$$

Indexes r_1, r_2, r_3 are randomly selected within the range $\{1, Np\}$ and they are pairwise different from each other and from index i.

Crossover generates a new vector $\mathbf{u}_{i,g+1}$:

$$u_{ji,g+1} = \begin{cases} v_{ji,g+1} & \text{if } rand(j) \leq Cr \text{ or } j = rn(i), \\ x_{ji,g} & \text{otherwise.} \end{cases} \tag{2}$$

$rand(j) \in [0, 1]$ is the j-th evaluation of the uniform random generator number. $rn(i) \in \{1, 2, \ \ldots, D\}$ is a randomly chosen index.

Selection performs a greedy selection scheme:

$$\mathbf{x}_{i,g+1} = \begin{cases} \mathbf{u}_{i,g+1} & \text{if } f(\mathbf{u}_{i,g+1}) \leq f(\mathbf{x}_{i,g}), \\ \mathbf{x}_{i,g} & \text{otherwise.} \end{cases} \tag{3}$$

The initial population is selected uniformly at random within the specified lower and upper bounds defined for each variable x_j. The algorithm iterates until the termination criterion is satisfied.

Subsequently, in addition to (1) and (2), other mutation and crossover strategies were proposed in literature [3–5]. Although the core DE has significant potential, modifications to its original structure were recognized as necessary to achieve substantial performance improvements.

2.2. Statistical Tests

DE belongs to the stochastic algorithms that can return a different solution in each run, as they use random generators. Therefore, we run the stochastic algorithms several times and statistically compare the obtained results. Based on this, we can then say that a given algorithm is statistically better (worse) than another algorithm with a certain level of significance.

Statistical tests designed for statistical analyses are classified into two categories: parametric and non-parametric. Parametric tests are based on assumptions that are most probably violated when analyzing the performance of stochastic algorithms based on computational intelligence. For this reason, in this paper, we use several non-parametric tests for pairwise and multiple comparisons.

To infer information about two or more algorithms based on the given results, two hypotheses are formulated: the null hypothesis H_0 and the alternative hypothesis H_1. The null hypothesis is a statement of no difference, whereas the alternative hypothesis represents a difference between algorithms. When using a statistical procedure to reject a hypothesis, a significance level α is set to determine the threshold at which the hypothesis can be rejected. Instead of predefining a significance level (α), the smallest significance level that leads to the rejection of H_0 can be calculated. This value, known as the p-value, represents the probability of obtaining a result at least as extreme as the observed one, assuming H_0 is true. The p-value provides insight into the strength of the evidence against H_0. The smaller the p-value, the stronger the evidence against H_0. Importantly, it achieves this without relying on a predetermined significance level.

The simple sign test for pairwise comparison assumes that if the two algorithms being compared have equivalent performance, the number of instances where each algorithm outperforms the other would be approximately equal. The Wilcoxon signed-rank test extends this idea by considering not only the number of wins but also the magnitude of differences. Specifically, it ranks the absolute differences in performance for each benchmark function and uses these ranks to determine whether the differences are statistically significant. This test is non-parametric, meaning it does not assume that the performance data follows a normal distribution, which makes it especially suitable for comparing optimization algorithms. This test takes the mean performance from multiple runs for each benchmark function of the two algorithms. Unlike parametric tests, where the null hypothesis H_0 assumes the equivalence of means, these tests define the null hypothesis (H_0) as the equivalence of medians. The test adds the ranks of the positive and negative differences separately. The smaller of the two rank sums is used to compute the test statistic. If the test finds that the sum of ranks for positive and negative differences is significantly different, the null hypothesis (that the two algorithms have equivalent median performance) is rejected. This suggests that one algorithm consistently outperforms the other.

The Friedman test is a non-parametric test used to detect differences in the performance of multiple algorithms across multiple benchmark functions. It is an alternative to the repeated-measures ANOVA when the assumptions of normality are not met. This makes the Friedman test well-suited for comparing optimization algorithms, whose performance data is often non-normal. In this test, the null hypothesis H_0 states that the medians of the different algorithms are equivalent. In contrast, the alternative hypothesis H_1 suggests that at least two algorithms have significantly different medians, which means that they differ significantly in performance. To perform the test, each algorithm's performance is ranked for every benchmark problem: the best-performing algorithm gets rank 1, the second-best gets rank 2, and so on. These ranks are assigned independently for each problem. Once all ranks are assigned, the test calculates the average rank of each algorithm across all problems. The Friedman test statistic is then computed based on these average ranks to assess whether the observed differences are greater than what would be expected by chance. If the result of the Friedman test is statistically significant, a post-hoc analysis (e.g., the Nemenyi test) is conducted to determine which specific pairs of algorithms differ. In the Nemenyi test, the critical distance (CD) is calculated. It is a threshold used to determine whether the difference in performance is statistically significant. If the difference between the average ranks of two algorithms exceeds this CD, their performance is considered significantly different.

The Mann-Whitney U-score test, which is also called the Wilcoxon rank-sum test, is another non-parametric statistical test used to compare two algorithms and determine whether one tends to have higher values than the other. Unlike the Wilcoxon signed-rank test, which compares two related or paired samples (e.g., same problems run by both algorithms), the U-score test is designed for independent samples, such as comparing the results of two algorithms across different trials or problem instances without assuming any pairing between results. The test ranks all results from both algorithms together. The smallest value receives rank 1, the next smallest rank 2, and so on, regardless of which algorithm the value came from. Once all values are ranked, the ranks are then separated back into their respective groups (i.e., by algorithm), and the sum of ranks for each algorithm is calculated. The test statistic (U-score) is derived from these rank sums and represents the number of times observations in one group precede observations in the other. A lower U-score value indicates a stronger difference between the two distributions. The test then assesses whether this observed difference in rank sums will likely have occurred by chance under the null hypothesis (H_0), which states that the two algorithms are drawn from the same distribution (i.e., neither consistently performs better).

Readers interested in a more detailed explanation of statistical tests are advised to consult the literature [6,7]. Our research focuses on the results at the end of the optimization process, while others also consider the convergence of their results, as it is also a desirable characteristic [8].

3. Related Works

One of the first DE surveys was published by Neri and Tirronen in 2010 [9], and another in 2011 by Das and Sugathan [10]. The first one includes some experimental results obtained with the algorithms available at that time. The second reviews the basic steps of the DE, discusses different modifications defined at that time, and also provides an overview of the engineering applications of the DE. Both surveys stated that DE will remain an active research area in the future. The time that followed confirmed this, as noted by the same authors who published the updated survey 5 years later [3]. Several years later, Opara and Arabas in [4] provided a theoretical analysis of DE and discussed the characteristics of DE genetic operators and an overview of the population diversity and dynamics models. In 2020, Pant et al. [5] published another survey of the literature on DE, which also provided a bibliometric analysis of DE. Some papers review specific components of DE. DE is very sensitive to its parameter settings; the authors in [11–14] reviewed parameter control methods. Piotrowski et al. [15] analyzed population size settings and adaptation. Parouha and Verma reviewed hybrids with DE [16]. Zhongqiang et al. [17] made an exhaustive listing of more than 500 nature-inspired meta-heuristic algorithms. They also empirically analyzed 11 newly proposed meta-heuristics with many citations, comparing them against four state-of-the-art algorithms, 3 of which were DE-based. Their results show that 10 out of 11 algorithms are less efficient and robust than four state-of-the-art algorithms; they are also inferior to them in terms of convergence speed and global search ability. Cited papers show that over time, DE has evolved in many different directions and remains one of the most promising optimization algorithms. The present paper analyzes the DE algorithms that were proposed very recently. The focus is on their mechanisms and performance.

The algorithms proposed in the literature are very diverse in terms of exploration and exploitation capabilities, so it is important how we evaluate and compare them. In [6], Derrac et al. give a practical tutorial on the use of statistical tests developed to perform both pairwise and multiple comparisons for multi-problem analysis. Traditionally, algorithms are evaluated using non-parametric statistical tests with the sign test, the Friedman test, the Mann-Whitney test, and both the Wilcoxon signed-rank and rank sum tests are among the most commonly used. They are based on ordinal results, like the final fitness values of multiple trials. Convergence speed is also an important characteristic of the optimization algorithm. A trial may also terminate upon reaching a predefined target value. In such cases, the outcome is characterized by both the time taken to achieve the target value (if achieved) and the final fitness value. The authors in [8] propose a way to make existing non-parametric tests to compare both the speed and accuracy of algorithms. They demonstrate trial-based dominance using the Mann-Whitney U-score test to compute U-scores with trial data from multiple algorithms. The authors demonstrate that U-scores are much more effective than traditional ways to rank solutions of algorithms. Non-parametric statistical methods are time-consuming and primarily focus on mean performance measures. To use them, raw results of algorithms are needed for comparison. Goula and Sotiropoulo [18] propose a multi-criteria decision-making technique to evaluate the performance of algorithms and rank them. Multiple criteria included best, worst, median, mean, and standard deviation of the error values. They employed equal weighting because the true weights of the criteria are generally unknown in practice.

To perform a comparative analysis of algorithms, the selection of problems on which the algorithms are evaluated is crucial [19]. During the last 20 years, several problem sets were defined. CEC'17 seems to be the most difficult, as it contains a low percentage of unimodal functions (7%), and a high percentage (70%) of hybrid or composition functions.

If we compare our research with related works, we would highlight the following: We conduct an in-depth comparison of the latest DE-based algorithms—particularly those proposed in 2024—and the basic DE algorithm. L-SHADE and jSO were included in the comparison, as they form the core of many improved DE-based algorithms. NL-SHADE-LBC was included as it is one of the best-performing DE-based algorithms in CEC'22 Competition. The other four are DE-based algorithms from the CEC'24 Competition. The analysis includes a wide range of dimensions: $D = 10$, $D = 30$, $D = 50$, and $D = 100$, and three statistical tests. To the best of our knowledge, such an analysis has not yet been performed.

4. Algorithms in the Comparative Study

In this section, we briefly describe the algorithms in comparison. We expose their main characteristics. Readers interested in detailed descriptions of the algorithms are encouraged to study the given references. All algorithms are based on the original DE.

4.1. L-SHADE

L-SHADE algorithm was proposed in 2014 [20]. Its denominator is the SHADE (Success-History based Adaptive DE) algorithm [21], proposed in 2013, which introduces external archive A into DE to enrich population diversity. The archive A saves target vectors if trial vectors improve them. SHADE algorithm also introduces a history-based adaptation scheme for adjusting F_i and Cr_i. Successful F_i and Cr_i are saved into historical memory by using the weighted Lehmer mean in each generation. The mutant vector is generated using a "DE/currect-to-pbest/1" mutation strategy using the external archive. In 2014, L-SHADE [20] added a linear population size reduction strategy to balance exploration and exploitation.

4.2. NL-SHADE-LBC

NL-SHADE-LBC (Non-Linear population size reduction Success-History Adaptive Differential Evolution with Linear Bias Change) algorithm was proposed in 2022 [22] as an improved variant of the NL-SHADE-RSP [23] approach, which integrates several key concepts, such as non-linear reduction of population size, rank-based selective pressure in the mutation strategy "DE/currect-to-pbest/1", automatic tuning of archive usage probability, and a set of rules for regulating the crossover rate. In contrast, NL-SHADE-LBC employs a fixed probability for archive usage and introduces a revised archive update strategy. It also applies a biased parameter adaptation approach using the generalized Lehmer mean for both the scaling factor F and the crossover rate Cr. The NL-SHADE-LBC uses a modified bound constraint handling technique. Whenever a mutant vector is generated outside the boundaries, it is generated again, with new parameters. The resampling procedure is repeated up to 100 times. If the new point still lies outside the boundaries, it is then handled using the midpoint target method.

4.3. jSO

jSO was proposed in 2017 [24]. It significantly improves the performance of basic DE. Its denominator is L-SHADE algorithm. The iL-SHADE [25] algorithm modifies its memory update mechanisms using an adjustment rule that depends on the evolution stage. Higher values for Cr and lower values for F were propagated at the early stages of evolution with the idea of enriching the population diversity. It also uses a linear increase in greediness factor p. jSO is based on iL-SHADE. It introduces a new parameter F_w to the scaling factor F to the mutation strategy for tuning the exploration and exploitation ability. A smaller factor F_w is applied during the early stages of the evolutionary process, while a larger factor F_w is utilized in the later stages. Using a parameter F_w, jSO adapts a new weighted version of the mutation strategy ”DE/currect-to-pbest-w/1”.

4.4. jSOa

jSOa (jSO with progressive Archive) algorithm was proposed in 2024 [26]. It proposes a more progressive update of the external archive A. In the SHADE algorithm, and subsequently in the jSO algorithm, when archive A reaches a predefined size, space for new individuals is created by removing random individuals. A newly proposed jSOa introduces a more systematic approach for storing outperformed parent individuals in the archive of the jSO algorithm. When the archive reaches its full capacity, its individuals are sorted based on their functional values, dividing the archive into two sections: better and worse individuals. The newly outperformed parent solution is then inserted into the "worse" section, ensuring that better solutions are preserved while the less effective individuals in the archive are refreshed. Notably, the new parent solution can still be inserted into the archive, even if it performs worse than an existing solution already in the archive. Using this approach, 50% of the better individuals in the archive are not replaced. Except for the archive, the other steps of the jSO algorithm remain unchanged in jSOa.

4.5. mLSHADE-RL

The mLSHADE-RL (multi-operator Ensemble LSHADE with Restart and Local Search) algorithm was also proposed in 2024 [27]. Its core algorithm is L-SHADE, which was extended into the LSHADE-EpSin algorithm with an ensemble approach to adapt the scaling factor using an efficient sinusoidal scheme [28]. Additionally, LSHADE-EpSin uses a local search method based on Gaussian Walks, which is used in later generations to improve exploitation. In LSHADE-EpSin, a mutation strategy DE/current-to-pbest/1 with an external archive is applied. LSHADE-cnEpSin [29] is the improved version of LSHADE-EpSin, which uses a practical selection for scaling parameter F and a covariance matrix learning with Euclidean neighborhoods to optimize the crossover operator. mLSHADE-RL further enhances LSHADE-cnEpSin algorithm. It integrates three mutation strategies such as ”DE/current-topbest-weight/1” with archive, ”DE/current-to-pbest/1” without archive, and ”DE/current-to-ordpbest-weight/1”. It has been shown that multi-operator DE approaches adaptively emphasize better-performing operators at various stages. mLSHADE-RL [27] also uses a restart mechanism to overcome the local optima tendency. It consists of two parts: detecting stagnating individuals and enhancing population diversity. Additionally, mLSHADE-RL applies a local search method in the later phase of the evolutionary procedure to enhance the exploitation capability.

4.6. RDE

RDE (Reconstructured Differential Evolution) algorithm was also proposed in 2024 [30]. Just like the previously introduced algorithms, RDE can also be understood as improving the L-SHADE algorithm. Like other algorithms, starting with SHADE, it uses an external archive. The research has demonstrated that incorporating multiple mutation strategies is beneficial; however, adding all-new strategies does not necessarily guarantee the best performance. RDE uses a hybridization of two mutation strategies, ”DE/current-to-pbest/1” and ”DE/current-to-order-pbest/1” strategies. The allocation of population resources between the two strategies is governed by an adaptive mechanism that considers the average fitness improvement achieved by each strategy. In RDE, parameters F and Cr are adapted similarly to how they are in jSO. In LSHADE-RSP, the use of a fitness-based rank pressure was proposed for the first time [31]. The idea is to correct the probability of different individuals being selected in DE, with rank indicators assigned to each of them. In RDE, the selection strategy from LSHADE-RSP was slightly modified, as it is based on three instead of two terms.

4.7. L-SRTDE

Like the previous three algorithms, L-SRTDE (Linear population size reduction Success RaTe-based Differential Evolution) was also proposed in 2024 [32]. It focuses on one of the most important parameters of the DE, the scaling factor. L-SRTDE adapts its value based on the success rate, which is defined as the ratio of improved solutions in each generation. The success rate also influences the parameter of the mutation strategy, which determines the number of top solutions from which the randomly chosen one is selected. Another minor modification in L-SRTDE is the usage of repaired crossover rate Cr. The idea is that instead of using the sampled crossover rate Cr for crossover, the actual Cr_a value is calculated as the number of replaced components divided by the dimensionality D. All other characteristics of the L-SRTDE algorithm are taken from the L-NTADE algorithm [33], which introduces the new mutation strategy, called ”r-new-to-ptop/n/t” and significantly alters the selection step.

5. Methodology

The purpose of the analysis is to evaluate the performance of recent DE-based algorithms. To evaluate their contributions, we include two well-established baselines and promising new algorithms. To maintain depth in the analysis without overwhelming complexity, we limited the selection of algorithms to recent CEC (Congress on Evolutionary Computation) competition. In addition to the four DE-based algorithms from CEC’24, the basic DE, L-SHADE, NL-SHADE-LBC, and jSO algorithms were used in comparison. L-SHADE is a denominator of many improved versions of DE. NL-SHADE-LBC ranked second in the CEC 2022 Special Session and Competition on Single-Objective Real-Parameter Numerical Optimization. jSO is incorporated in the comparison, as it ranked in second place in the CEC 2017 Special Session and Competition on Single-Objective Real Parameter Numerical Optimization. jSO is also a highly cited algorithm, as many authors use it as a baseline algorithm for further development.

The selected algorithms were analyzed separately for each problem dimension: 10, 30, 50, and 100. They were run on all test problems a predetermined number of times. The best error value for every $10 \cdot D$ evaluations was recorded for each run. The run was finished after a maximum number of function evaluations had been reached. The maximum number of evaluations was determined based on the problem’s dimensionality and is the same for all algorithms.

Having raw results for all algorithms, five statistical measures were calculated: best, worst, median, mean, and standard deviation of the error values. The algorithms were evaluated using the following statistical tests: Wilcoxon signed-ranks test, Friedman test, and Mann-Whitney U-score test. It is important to note that the computation of the U-score in this article differs from the U-score used in the CEC 2024 competition [8,34]. In our study, we applied the classic Mann-Whitney U-score test, whereas the U-score in the CEC 2024 competition incorporates the "target-to-reach" value. Convergence speed was also analyzed using convergence plots.

6. Experiments

We conducted experiments in which we empirically evaluated the progress brought by the improvements to the DE algorithm over the past year. We included the following algorithms: basic DE, L-SHADE [20], NL-SHADE-LBC [22], jSO [24], jSOa [26], mLSHADE-RL [27], RDE [30], and L-SRTDE [32]. For DE we used the following setting: $F = 0.5$, $Cr = 0.9$ and $Np = 100$, which remains fixed during the optimization process. In contrast, other algorithms use a population size reduction mechanism and employ self-adaptive or adaptive mechanisms to control parameters. Each algorithm includes additional parameters specific to its design. We used their default configurations, as described in their original publications, and did not alter any of these parameter settings. Although the authors of the algorithms have published some results; we conducted the runs ourselves in the study based on the publicly available source code of the algorithms. The experiments were performed following the CEC'24 Competition [35] instructions: experiments with the dimensions $D = 10$, $D = 30$, $D = 50$, and $D = 100$ were done. Each algorithm was run 25 times for each test function. With 25 runs, it becomes possible to estimate central tendencies (such as the median or mean) and dispersion (such as variance) with reasonable accuracy, while also ensuring that statistical tests (e.g., the Wilcoxon signed-rank test, Mann-Whitney U-score test, and Friedman test) have sufficient power to detect meaningful differences between algorithms. Each run stopped either when the error obtained was less than 10^{-8} or when the maximal number of evaluations $Max_{FEs} = 10000 \cdot D$ was achieved.

6.1. CEC'24 Benchmark Suite

CEC'24 Special Session and Competition on Single Objective Real Parameter Numerical Optimization [35] was based on the CEC'17 benchmark suite, which is a collection of 29 optimization problems that are based on shifted, rotated, non-separable, highly ill-conditioned, and complex functions. The benchmark problems aim to replicate the behavior of real-world problems, which often exhibit complex features that basic optimization algorithms may struggle to capture. The functions are:

- unimodal functions: Bent Cigar function and Zakharov function. Unimodal functions are theoretically the simplest and should present only moderate challenges for well-designed algorithms.
- simple multimodal functions: Rosenbrock's function, Rastrigin's function, expanded Scaffer's F6 function, Lunacek Bi_Rastrigin function, non-continuous Rastrigin's function, Levy function, and Schwefel's function. Simple multimodal functions, characterized by multiple optima, are rotated and shifted. However, their fitness landscape often exhibits a relatively regular structure, making them suitable for exploration by various types of algorithms.
- hybrid functions formed as the sum of different basic functions: Each function contributes a varying weighted influence to the overall problem structure across different regions of the search space. Consequently, the fitness landscape of these functions often varies in shape across different areas of the search space.
- composition functions formed as a weighted sum of basic functions plus a bias according to which component optimum is the global one: They are considered the most challenging for optimizers because they extensively combine the characteristics of sub-functions and incorporate hybrid functions as sub-components.

The search range is $[-100, 100]^D$. The functions are treated as black-box problems. Table 1 presents function groups.

Table 1. Function groups.

Function group	Functions
Unimodal functions	f1, f2
Multimodal functions	f3, f4, f5, f6, f7, f8, f9
Hybrid functions	f10, f11, f12, f13, f14, f15, f16, f17, f18, f19
Composition functions	f20, f21, f22, f23, f24, f25, f26, f27, f28, f29

6.2. Results

In this subsection, we present experimental results using three metrics for a comparison of algorithms, namely the U-score, the Wilcoxon test, and the Friedman test. The experimental results were obtained on 29 benchmark functions for eight selected algorithms (DE, L-SHADE, NL-SHADE-LBC, jSO, jSOa, mLSHADE-RL, RDE, and L-SRTDE) on four dimensions (10, 30, 50, and 100). The main experimental results are summarized in the rest of this subsection, while additional detailed and collected results are placed in the Supplementary Materials:

- Tables S1–S32 present the best, worst, median, mean, and standard deviation values after maximal number of function evaluations for eight algorithms on 29 benchmark functions. For dimension $D = 10$, DE solved 9 functions; jSO and mLSHADE-RL each solved 11; NL-SHADE-LBC and L-SHADE solved 12; jSOa solved 13; while RDE and L-SRTDE solved 14 functions. At $D = 30$, DE solved three functions; NL-SHADE-LBC, L-SHADE, jSO, and jSOa each solved four; mLSHADE-RL and RDE solved five; and L-SRTDE achieved the best performance with eight functions solved. For $D = 50$, DE solved only two functions; NL-SHADE-LBC managed one; L-SHADE, jSO, jSOa, mLSHADE-RL, and RDE each solved four; and L-SRTDE solved five functions. At the highest dimension, $D = 100$, DE did not solve any function; NL-SHADE-LBC and L-SRTDE each solved one; L-SHADE, jSO, mLSHADE-RL, and RDE each solved two; and jSOa performed best with three functions solved.
- Convergence speed graphs are depicted in Figures S1–S16. L-SRTDE demonstrates the best performance, exhibiting the steepest and most consistent convergence curves. RDE also performs competitively, while the remaining algorithms generally take longer to reach lower objective values. DE, on the other hand, shows minimal improvement throughout the runs, maintaining consistently high objective values.

6.2.1. U-score

Tables 2–5 show the U-score values for eight algorithms on 29 benchmark functions for dimensions $D = 10$, $D = 30$, $D = 50$, and $D = 100$, respectively. For each function, the rank of each algorithm is presented in red. At the bottom of the table, a sum of the U-score values and a sum of ranks, RS, are shown.

Overall U-score values for each dimension are collected in Table 6. In this table, ”Total” indicates a sum of the U-scores values over all dimensions (higher value is better), while ”totalRS” summarizes ranks of algorithms (lower value is better). Based on these results, we can see that L-SRTDE has the first rank, followed by RDE, jSOa, jSO, L-SHADE, mLSHADE-RL, NL-SHADE-LBC, and DE. Table 7 presents the overall ranks of the algorithms for each dimension separately, as well as the cumulative rank across all dimensions, based on the U-score. L-SRTDE holds rank one across all dimensions, while RDE holds rank 2 in all dimensions except for D = 10. jSO and jSOa alternate between ranks 3 and 4. L-SHADE consistently holds rank five across all dimensions. NL-SHADE-LBC and mLSHADE-RL alternate between ranks 6 and 7. DE consistently holds the lowest rank, which is eight, across all dimensions.

Table 2. U-scores of algorithms for 10D. Sum denotes the total of U-scores, and RS represents the sum of ranks across all 29 functions.

Func.	DE	NL-SHADE-LBC	L-SHADE	jSO	jSOa	mLSHADE-RL	RDE	L-SRTDE
f1	2187.5/4.5	2187.5/4.5	2187.5/4.5	2187.5/4.5	2187.5/4.5	2187.5/4.5	2187.5/4.5	2187.5/4.5
f2	2187.5/4.5	2187.5/4.5	2187.5/4.5	2187.5/4.5	2187.5/4.5	2187.5/4.5	2187.5/4.5	2187.5/4.5
f3	1906/7	1794/8	2300/3.5	2300/3.5	2300/3.5	2300/3.5	2300/3.5	2300/3.5
f4	0/8	2264.5/6	1155.5/7	2685/3	2425.5/5	2749.5/2	2508/4	3712/1
f5	2212.5/4	2212.5/4	2212.5/4	2212.5/4	2212.5/4	2212.5/4	2212.5/4	2012.5/8
f6	0/8	2619/3	2157/5	2540/4	2060/6	2917/2	1771/7	3436/1
f7	0/8	2714/3	1300.5/7	2377/4	2270/5	2785.5/2	2194/6	3859/1
f8	2187.5/4.5	2187.5/4.5	2187.5/4.5	2187.5/4.5	2187.5/4.5	2187.5/4.5	2187.5/4.5	2187.5/4.5
f9	0/8	2834/2	2298/4	2137.5/5	2304/3	2081.5/7	2084/6	3761/1
f10	1037.5/8	2412.5/2.5	2304/6	2412.5/2.5	2412.5/2.5	2200.5/7	2412.5/2.5	2308/5
f11	2609/4	1884/7	2082/6	2817/2	2881.5/1	124.5/8	2734/3	2368/5
f12	1944/7	2581/3	2307.5/4	2599/2	2173/5	1089.5/8	1946.5/6	2859.5/1
f13	2134/6	2750/2	1232/7	2655/5	2750/2	563/8	2750/2	2666/4
f14	3872/1	1939/5	2606/2	1770/7	1886/6	835/8	2013/4	2579/3
f15	3064/2	1137/8	2715/3	1600/7	3290/1	2075/4	1627.5/6	1991.5/5
f16	797/8	1447/7	3872/1	2336/4	1961/5	2618/3	2902/2	1567/6
f17	3820/1	1804/7	2710/2	2191.5/6	2369.5/3	24/8	2258/5	2323/4
f18	2592.5/4	1390.5/7	2242.5/5	2761.5/3	3420.5/1	275.5/8	2813/2	2004/6
f19	1556/7	2676/3	3263/2	498/8	3375/1	1730/6	2159/5	2243/4
f20	1452/8	2494/2	2003/6	2382.5/5	2396/4	2471/3	1684.5/7	2617/1
f21	790/8	2362.5/4	2362.5/4	2362.5/4	2362.5/4	2362.5/4	2262/7	2635.5/1
f22	189/8	3164.5/1	889.5/7	2837.5/4	2528/5	1795.5/6	3159.5/2	2936.5/3
f23	1622/7	2969.5/1	1197/8	2207.5/5	1714.5/6	2864.5/2	2263/4	2662/3
f24	2882/1	2676/2	2605/3	2329/5	2517/4	2006/6	1156/8	1329/7
f25	2187.5/4.5	2187.5/4.5	2187.5/4.5	2187.5/4.5	2187.5/4.5	2187.5/4.5	2187.5/4.5	2187.5/4.5
f26	1856/6	425/7	1971/4.5	1971/4.5	2804/3	4318/1	3803/2	352/8
f27	2437.5/1.5	2235.5/4	2233.5/5	2224.5/6	1614.5/8	2068.5/7	2248.5/3	2437.5/1.5
f28	411/8	2312/5	2347/4	1388/6	1382/7	3276/2	3062.5/3	3321.5/1
f29	1035/7	1643/6	2533/4	2946/2	2931/3	681/8	3649/1	2082/5
sum/RS	48969/163.5	63491/127.5	63649/132	65291/129.5	69090.5/116	57174/145.5	68723/123	71112.5/107
Rank	8	6	5	4	2	7	3	1

Table 3. U-scores of algorithms for 30D. Sum denotes the total of U-scores, and RS represents the sum of ranks across all 29 functions.

Func.	DE	NL-SHADE-LBC	L-SHADE	jSO	jSOa	mLSHADE-RL	RDE	L-SRTDE
f1	2187.5/4.5	2187.5/4.5	2187.5/4.5	2187.5/4.5	2187.5/4.5	2187.5/4.5	2187.5/4.5	2187.5/4.5
f2	0/8	2500/4	2500/4	2500/4	2500/4	2500/4	2500/4	2500/4
f3	1864.5/6	1241/7	2087.5/4	2087.5/4	2087.5/4	3836/1	3625/2	671/8
f4	0/8	811/7	3389/2	2655/4	2057/5	1178/6	3059/3	4351/1
f5	1577.5/8	1952.5/6	2470/3	2365/4	2575/1.5	2250/5	2575/1.5	1735/7
f6	0/8	859/7	3564/2	2681/4	2170/5	1158/6	3000/3	4068/1
f7	0/8	939/7	3200/3	2520/4	1969/5	1154/6	3374.5/2	4343.5/1
f8	2187.5/4.5	2187.5/4.5	2187.5/4.5	2187.5/4.5	2187.5/4.5	2187.5/4.5	2187.5/4.5	2187.5/4.5
f9	0/8	625/7	2576/3	2400/6	2657/2	2408/5	2468/4	4366/1
f10	384/8	1644/5	1595/6	3222.5/2	2987.5/3	732/7	2979.5/4	3955.5/1
f11	0/8	1569/5	1383/6	3194/3	3317/2	940/7	2745/4	4352/1
f12	384/7	2152/6	2586/3	2583/4	2161/5	366/8	3324/2	3944/1
f13	573/7	1337/6	2287/5	2791/4	3179/2	52/8	3001/3	4280/1
f14	625/7	1596/6	2243/4	3167/3	2203/5	0/8	3522/2	4144/1
f15	363/8	1496/7	1923/6	2124/3	2006/4	1998/5	3386/2	4204/1
f16	9/8	1114/7	2794/2	2323/5	2634/3	2020/6	2426/4	4180/1
f17	641/7	1569/6	2052/5	2979/4	3606/1	1/8	3302/3	3350/2
f18	694/7	1434/6	2063/5	2861/3	2667/4	0/8	3523/2	4258/1
f19	2619/3	351/8	2094/6	2503/4	2326/5	985/7	2704/2	3918/1
f20	0/8	916/7	3065/3	2695/4	1901/5	1308/6	3276/2	4339/1
f21	2200/4	2200/4	2200/4	2200/4	2200/4	2100/8	2200/4	2200/4
f22	0/8	2691/3	2311/5	2679/4	1706/6	1056/7	2821/2	4236/1
f23	0/8	3677/2	1711/6	2187/5	1304/7	2217/4	2350/3	4054/1
f24	1005.5/6	632/8	836/7	2566.5/5	2602/4	2706/3	4372/1	2780/2
f25	335/8	2652/3	2169/6	2270/4	2249/5	815/7	2885/2	4125/1
f26	2506/3	1375/6	588.5/8	2133/4	1347.5/7	1630/5	3873/2	4047/1
f27	2065/7	2177.5/6	1383/8	2239.5/5	2321.5/4	2442.5/1	2435/3	2436/2
f28	302/8	534/7	2323/5	2499/4	2554/3	1577/6	3788/2	3923/1
f29	1926/5	386/8	1615/7	2472/3	2830/2	2128/4	4372/1	1771/6
sum/RS	24448.5/198	44805/170	63383/137	73272/116	68492/116.5	43932.5/165	88261/78.5	100906/63
Rank	8	6	5	3	4	7	2	1

Table 4. U-scores of algorithms for 50D. Sum denotes the total of U-scores, and RS represents the sum of ranks across all 29 functions.

Func.	DE	NL-SHADE-LBC	L-SHADE	jSO	jSOa	mLSHADE-RL	RDE	L-SRTDE
f1	142/8	483/7	2850/3	2850/3	2850/3	2625/6	2850/3	2850/3
f2	0/8	625/7	2825/3	2825/3	2825/3	2750/6	2825/3	2825/3
f3	1165/7	539/8	1526.5/6	2083/5	2142.5/4	4182/1	2787/3	3075/2
f4	0/8	629/7	3556/2	2609/4	1978/5	1351/6	3002/3	4375/1
f5	3151.5/3	1469/6	2366/4	2046/5	3365/2	0/8	3812.5/1	1290/7
f6	0/8	758/7	3532/2	2793/4	1869/5	1573/6	3440/3	3535/1
f7	0/8	644/7	3524/2	2491/4	2106/5	1300/6	3066/3	4369/1
f8	2150.5/7	2512.5/3.5	2512.5/3.5	2512.5/3.5	2512.5/3.5	274.5/8	2512.5/3.5	2512.5/3.5
f9	0/8	625/7	2560/3	2390/6	2481/5	2578/2	2491/4	4375/1
f10	218/8	1597/5	1407/6	2770.5/4	2956.5/3	547/7	3803/2	4201/1
f11	4/8	693/7	2003/6	2692/4	2748/3	2006/5	2979/2	4375/1
f12	434/7	1146/6	2347/5	3102/2	2797/4	307/8	3009/3	4358/1
f13	623/7	1275/6	2003/5	3001/3	3478/2	2/8	2867/4	4251/1
f14	625/7	1568/6	1643/5	2771/4	3358/2	0/8	3179/3	4356/1
f15	79/8	1618/6	1880/5	2067/4	1603/7	2459/3	3430/2	4364/1
f16	36/8	1286/6	2671/3	2365/4	2281/5	1135/7	3574/2	4152/1
f17	43/8	1545/5	1535/6	3253/2	3171/3	683/7	2947/4	4323/1
f18	600/7	1348/6	1749/5	2863/4	3514/2	65/8	2986/3	4375/1
f19	40/8	1209/7	1628/6	1841/5	2726/3	2396/4	3285/2	4375/1
f20	0/8	673/7	3461/2	2712/4	1735/5	1474/6	3088/3	4357/1
f21	1408/6	3375/1.5	1057/8	2247/4	1329/7	2841/3	1868/5	3375/1.5
f22	0/8	1751/6	2465/4	2532/3	2083/5	1086/7	3214/2	4369/1
f23	0/8	3597/2	2022/5	2334/4	1275/6	955/7	2984/3	4333/1
f24	1493.5/6	373/8	2035/5	2483/4	2975.5/3	774/7	3657/2	3709/1
f25	31/8	1619/6	2468/4	2840/3	2220/5	647/7	3300/2	4375/1
f26	2286/4	1473/6	893/7	2464/3	2281/5	10/8	3857/2	4236/1
f27	2543/4	616/8	2255/5	2900/2.5	2900/2.5	661/7	4350/1	1275/6
f28	491/7	193/8	2574/3	2113/5	2452/4	1668/6	4163/1	3846/2
f29	2173/4	628/7	586/8	1892/5	1528/6	3662/2	4375/1	2656/3
sum/RS	19736.5/204	35867.5/179	63934/131.5	73842/111	71540/118	40011.5/174	93701/75.5	108868/51
Rank	8	7	5	3	4	6	2	1

Table 5. U-scores of algorithms for 100D. Sum denotes the total of U-scores, and RS represents the sum of ranks across all 29 functions.

Func.	DE	NL-SHADE-LBC	L-SHADE	jSO	jSOa	mLSHADE-RL	RDE	L-SRTDE
f1	327/8	365/7	3425/2.5	3425/2.5	3425/2.5	1925/5	3425/2.5	1183/6
f2	0/8	625/7	2287/5	3216/3	4375/1	3436/2	2311/4	1250/6
f3	1359/7	821/8	1911.5/6	2150/3	2112.5/4	4268/1	2949/2	1929/5
f4	0/8	701/7	3293/2	2635/4	2208/5	1180/6	3112/3	4371/1
f5	2010/5	1939/6	798/7	2500/4	4309/1	0/8	3076/2	2868/3
f6	0/8	650/7	3438/2	2824/4	2851/3	2121/5	1622/6	3994/1
f7	0/8	677/7	3350/2	2574/4	2495/5	1215/6	2824.5/3	4364.5/1
f8	652.5/7	1978/5	1855.5/6	2926.5/4	3362.5/2	0/8	3362.5/2	3362.5/2
f9	0/8	626/7	2007/5	2825/3	2800/4	3048/2	1819/6	4375/1
f10	628/7	1730/5	1362/6	3112/3	3391/2	37/8	2865/4	4375/1
f11	90/8	535/7	1601/6	2205/5	2550/3	3848/2	2427/4	4244/1
f12	739/7	401/8	1832/5	3169/3	2542/4	839/6	3624/2	4354/1
f13	0/8	1834/5	1168/6	3151/3	3379/2	748/7	2845/4	4375/1
f14	1040/8	1152/7	1303/6	3166/2	2798/3	1444/5	2262/4	4335/1
f15	0/8	1819/5	1848/4	1711/6	1636/7	3686/2	2425/3	4375/1
f16	0/8	1396/7	2455/4	1661/6	1899/5	2585/3	3129/2	4375/1
f17	0/8	1753/6	1978/5	2724/3	3208/2	1470/7	1992/4	4375/1
f18	379/8	1464/5	966/7	3063/3	3538/2	1202/6	2513/4	4375/1
f19	0/8	1011/7	1466/6	2122/5	2330/4	3374/2	2822/3	4375/1
f20	0/8	939/7	3008/3	2670/4	2309/5	977/6	3222/2	4375/1
f21	0/8	4332/1	1223/7	2020/5	2264/4	2599/3	1269/6	3793/2
f22	744/7	1047/6	2541/5	2648/4	3301/2	125/8	2832/3	4262/1
f23	157/8	3573/2	1519/6	2476/4	1892/5	475/7	3046/3	4362/1
f24	1967/6	691/8	1464/7	2285/3	2113/5	2205/4	2506/2	4269/1
f25	652/7	1891/5	1563/6	2952/3	2571/4	113/8	3383/2	4375/1
f26	2222/5	1094/6	845/7	2625/3	2594/4	0/8	4350/1	3770/2
f27	970/7	155/8	2295/5	2481/3	2211/6	2447/4	3525/1	3416/2
f28	89/8	1095/7	2361/4	2400/3	2023/5	1514/6	3820/2	4198/1
f29	464/8	707/7	2418/3	2403/4	1802/5	1632/6	4375/1	3699/2
sum/RS	14489.5/217	37001/180	57581/145.5	76119.5/106.5	78289/106.5	48513/151	83733/87.5	111774/50
Rank	8	7	5	4	3	6	2	1

Table 6. Overall U-scores of algorithms. Total denotes the sum of U-scores, and totalRS represents the sum of RS across all dimensions.

Dimension	DE	NL-SHADE-LBC	L-SHADE	jSO	jSOa	mLSHADE-RL	RDE	L-SRTDE
D10/RS	48969/163.5	63491/127.5	63649/132	65291/129.5	69090.5/116	57174/145.5	68723/123	71112.5/107
D30/RS	24448.5/198	44805/170	63383/137	73272/116	68492/116.5	43932.5/165	88261/78.5	100906/63
D50/RS	19736.5/204	35867.5/179	63934/131.5	73842/111	71540/118	40011.5/174	93701/75.5	108868/51
D100/RS	14489.5/217	37001/180	57581/145.5	76119.5/106.5	78289/106.5	48513/151	83733/87.5	111774/50
Total/totalRS	107644/782.5	181164/656.5	248547/546	288524/463	287412/457	189631/635.5	334418/364.5	392660/271

Table 7. Overall rank of algorithms based on the U-score.

Dimension	DE	NL-SHADE-LBC	L-SHADE	jSO	jSOa	mLSHADE-RL	RDE	L-SRTDE
D10	8	6	5	4	2	7	3	1
D30	8	6	5	3	4	7	2	1
D50	8	7	5	3	4	6	2	1
D100	8	7	5	4	3	6	2	1
Overall rank	8	7	5	3	4	6	2	1

We also analyzed the performance of the algorithms across different function families (see Table 1). To achieve this, we sum the U-score values of the algorithms for each function family and identify the best-performing algorithm for each function family across different problem dimensions. The obtained results are collected in Table 8. Unimodal functions are typically easier to solve. For dimension 10, all algorithms perform equally well. In higher dimensions (D = 50, 100) best performers are jSOa, jSO, L-SHADE, RDE, and L-SRTDE, which all perform similarly well. DE and NL-SHADE-LBC fall behind significantly as the dimension increases, especially at D = 50 and 100. Overall winner is jSOa, with the highest cumulative U-score (22,537.5), indicating strong and consistent performance across all dimensions. L-SRTDE, despite leading in many benchmarks, underperforms slightly in this group (17,170.5), mainly due to lower scores at D = 100. For multimodal functions, L-SRTDE achieves the best results, showing excellent ability to escape local optima and converge effectively. RDE (75,422) also dominates here. DE and NL-SHADE-LBC perform poorly, with much lower totals (22,424 and 39,809, respectively), confirming their difficulty with complex landscapes. jSO, jSOa, and L-SHADE are competitive but consistently fall behind L-SRTDE and RDE in all dimensions. On hybrid functions, L-SRTDE leads significantly (135,343), showing exceptional adaptability. RDE again ranks second (101,431), indicating strong hybrid problem-solving capability. DE and mLSHADE-RL perform weakly (33,028.5 and 43,945.5, respectively), suggesting challenges in handling mixed landscapes. jSO and jSOa improve in higher dimensions but still fall short of L-SRTDE and RDE. Composition functions are the most complex. L-SRTDE (117,833.5) and RDE (113,761) once again show superior performance, likely due to better exploration/exploitation balance. Moderate performers are jSO, jSOa, and L-SHADE with scores between 66,000 to 86,000. DE and NL-SHADE-LBC again show the lowest cumulative scores (41,440 and 64,510).

Based on U-score tests, we can conclude that L-SRTDE is the most consistently dominant algorithm across all function groups and dimensions, excelling particularly in multimodal, hybrid, and composition functions, which are typically more difficult. RDE is also a strong overall performer, often second to L-SRTDE. jSO and jSOa perform well on unimodal and hybrid functions, but lag in composition problems. DE shows consistently poor performance across all categories, especially as dimensionality increases. NL-SHADE-LBC and mLSHADE-RL are mid-tier at best and struggle with higher-dimensional and complex functions.

6.2.2. U-score-CEC'24

The obtained results of all eight algorithms were further evaluated using the U-score metric, which was used in CEC'24 [35]. The metric is referred to as U-scores-CEC'24. Sum of U-scores-CEC'24 and the sum of RS over all dimensions are collected in Table 9. We can see that the difference between the algorithms based on the U-score-CEC'24 is smaller than with the Mann–Whitney U-score (see, for example, the difference between DE and L-SRTDE). Overall ranks of algorithms based on the U-score-CEC'24 are given in Table 10. L-SRTDE and RDE once again occupy the top two positions, while DE remains last. The remaining algorithms appear in a slightly different order.

Table 8. U-scores arranged according to functions' families.

Function group	Dimension	DE	NL-SHADE-LBC	L-SHADE	jSO	jSOa	mLSHADE-RL	RDE	L-SRTDE
Unimodal functions	D10	4375	4375	4375	4375	4375	4375	4375	4375
	D30	2187.5	4687.5	4687.5	4687.5	4687.5	4687.5	4687.5	4687.5
	D50	142	1108	5675	5675	5675	5375	5675	5675
	D100	327	990	5712	6641	7800	5361	5736	2433
	Total	7031.5	11160.5	20449.5	21378.5	22537.5	19798.5	20473.5	17170.5
Multimodal functions	D10	6306	16625.5	13611	16439.5	15759.5	17233.5	15257	21268
	D30	5629.5	8615	19474	16896	15703	14171.5	20289	21722
	D50	6467	7176.5	19577	16924.5	16454	11258.5	21111	23531.5
	D100	4021.5	7392	16653	18434.5	20138	11832	18765	25264
	Total	22424	39809	69315	68694.5	68054.5	54495.5	75422	91785.5
Hybrid functions	D10	22388.5	17608.5	23030	19228	24106.5	9334.5	21203	20601
	D30	5908	12618	19425	24525	24099	6362	27933	36630
	D50	2484	11688	17459	23955	25676	9053	28256	38929
	D100	2248	11365	14617	22972	23880	19196	24039	39183
	Total	33028.5	53279.5	74531	90680	97761.5	43945.5	101431	135343
Composition functions	D10	13410	19975.5	18326	20453.5	20041	21559.5	23791	19943.5
	D30	10339.5	16324.5	15136.5	21246	19114	16671.5	29096	29572
	D50	10425.5	13625	16355	21805	19043.5	12304	31768	32174
	D100	7265	14585	16229	22290	20771	11110	29106	36144
	Total	41440	64510	66046.5	85794.5	78969.5	61645	113761	117833.5

Table 9. Overall U-scores-CEC'24 [35] of algorithms. Total denotes the sum of U-scores, and totalRS represents the sum of RS across all dimensions.

Dimension	DE	NL-SHADE-LBC	L-SHADE	jSO	jSOa	mLSHADE-RL	RDE	L-SRTDE
D10/RS	47469/161	69703/113	58466/133	45841/161	42809/173	67845/173	73864/112	101503/63
D30/RS	40530/166	59120/140	76171/107	58745/141	51148/153	56844/138	70204/120	94738/79
D50/RS	33101/179	54938/145	66733/125	65118/132	63267/136	64095/127	67809/119	92439/81
D100/RS	33301/182	49109/155	63294/138	77414/106	75190/104	61741/134	70408/121	77043/104
Total/totalRS	154401/688	232870/553	264664/503	247118/540	232414/566	250525/527	282285/472	365723/327

Table 10. Overall rank of algorithms based on the U-score-CEC'24 [35].

Dimension	DE	NL-SHADE-LBC	L-SHADE	jSO	jSOa	mLSHADE-RL	RDE	L-SRTDE
D10	6.5	3	5	6.5	8	4	2	1
D30	8	5	2	6	7	4	3	1
D50	8	7	3	5	6	4	2	1
D100	8	7	6	3	1.5	5	4	1.5
Overall rank	8	6	3	5	7	4	2	1

6.2.3. Wilcoxon Signed-rank Test

The Wilcoxon signed-rank test is used to compare two algorithms, and the number of all pair-wise comparisons for eight algorithms per one dimension is $8 \times 5/2 = 20$. Altogether, we have 80 comparisons. Usually, we only compare the newly designed algorithm with other algorithms, which reduces the number of comparisons. We chose L-SRTDE to compare it with other algorithms and the summarized results are presented in Table 11. For each dimension, the comparison using the Wilcoxon signed-rank test with $\alpha = 0.05$ of the L-SRTDE algorithm with other algorithms, and the summarized results are presented as $+/\approx/-$. Sign '$+$' means that L-SRTDE performs significantly better than the compared algorithm, '$-$' means that L-SRTDE performs significantly worse than the compared algorithm, while $\approx$ means that there is no significant difference between the compared algorithms. Detailed results are shown in Tables S33, S34, S35, and S36, while summarized results are collected in Table 11. The lines labeled 'Total' contain the sum of the values for each individual algorithm according to the dimension.

Table 11. The Wilcoxon signed-rank results of L-SRTDE versus other algorithms ($\alpha = 0.05$). ($+$ means that L-SRTDE performs significantly better than a compared algorithm.)

Dimension	Algorithm	$+/\approx/-$
10	DE	13/10/6
	NL-SHADE-LBC	9/18/2
	L-SHADE	8/17/4
	jSO	7/17/5
	jSOa	8/15/6
	mLSHADE-RL	11/16/2
	RDE	6/19/4
	Total	62/112/29
30	DE	22/7/0
	NL-SHADE-LBC	22/7/0
	L-SHADE	22/6/1
	jSO	19/9/1
	jSOa	19/8/2
	mLSHADE-RL	20/7/2
	RDE	17/8/4
	Total	141/52/10
50	DE	27/1/1
	NL-SHADE-LBC	26/3/0
	L-SHADE	25/3/1
	jSO	23/5/1
	jSOa	23/5/1
	mLSHADE-RL	25/2/2
	RDE	20/3/6
	Total	169/22/12
100	DE	28/1/0
	NL-SHADE-LBC	28/0/1
	L-SHADE	26/1/2
	jSO	25/2/2
	jSOa	24/2/3
	mLSHADE-RL	26/0/3
	RDE	21/2/6
	Total	178/8/17

We can see that L-SRTDE performs well but not overwhelmingly better in low-dimensional problems; There's greater performance overlap with competitors. L-SRTDE becomes more clearly superior in $D = 30$. Most differences are now statistically significant, especially against older or less adaptive methods. As the problem dimension increases, L-SRTDE pulls further ahead of its peers, with most comparisons resulting in statistically significant superiority. In a high-dimensional setting $D = 100$, L-SRTDE is consistently and significantly better than all other algorithms, solidifying its position as the best performer.

6.2.4. Friedman Test

As a third test to compare the algorithms, we used the Friedman test. The obtained results for each dimension are presented in Figures 1–4 and and Tables 12–15.

Table 12. Algorithms' ranks for 10D.

Algorithm	MeanRank	Rank
DE	4.7413793	6
NL-SHADE-LBC	4.5000000	4
L-SHADE	4.6551724	5
jSO	4.4482759	3
jSOa	4.1034483	2
mLSHADE-RL	4.8448276	8
RDE	4.8275862	7
L-SRTDE	3.8793103	1
$\chi^2 = 5.0974,\ p = 0.64808$		

Table 13. Algorithms' ranks for 30D.

Algorithm	MeanRank	Rank
DE	6.6896552	8
NL-SHADE-LBC	5.7241379	7
L-SHADE	4.8275862	5
jSO	4.1724138	3
jSOa	4.1896552	4
mLSHADE-RL	5.6724138	6
RDE	2.7413793	2
L-SRTDE	1.9827586	1
$\chi^2 = 95.3302,\ p = 9.911\text{e-}18$		

Table 14. Algorithms' ranks for 50D.

Algorithm	MeanRank	Rank
DE	7.2758621	8
NL-SHADE-LBC	6.1724138	7
L-SHADE	4.2931034	5
jSO	3.7758621	3
jSOa	3.9482759	4
mLSHADE-RL	6.0689655	6
RDE	2.7413793	2
L-SRTDE	1.7241379	1
$\chi^2 = 123.1061,\ p = 1.7262\text{e-}23$		

Table 15. Algorithms' ranks for 100D.

Algorithm	MeanRank	Rank
DE	7.5172414	8.0
NL-SHADE-LBC	6.1379310	7.0
L-SHADE	4.9827586	5.0
jSO	3.6724138	3.5
jSOa	3.6724138	3.5
mLSHADE-RL	5.0689655	6.0
RDE	3.1896552	2.0
L-SRTDE	1.7586207	1.0
$\chi^2 = 111.5434,\ p = 4.3921\text{e-}21$		

From the Tables 12–15 we can conclude that L-SRTDE consistently ranks 1st across all dimensions, demonstrating dominance that is not affected by problem dimensionality. RDE secures 2nd place in every case except for D = 10, where it drops to the penultimate position. jSO and jSOa alternate between 3rd and 4th place, while L-SHADE consistently holds 5th. mLSHADE-RL follows in 6th, with NL-SHADE-LBC in 7th. DE finishes last across all dimensions, confirming that more modern adaptive or hybrid algorithms clearly outperform it.

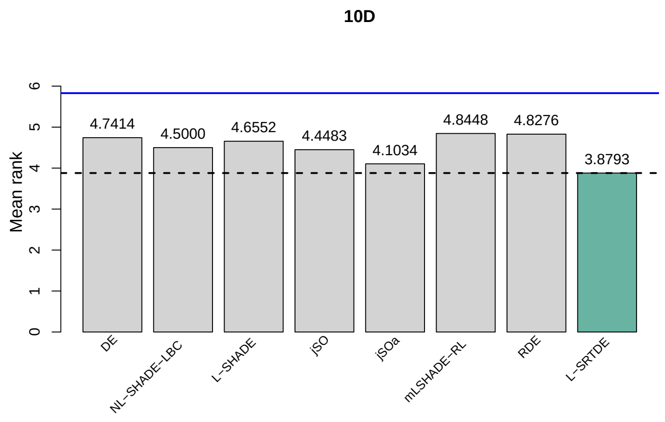

Figure 1. Friedman ranks of algorithms with a critical distance for 10D.

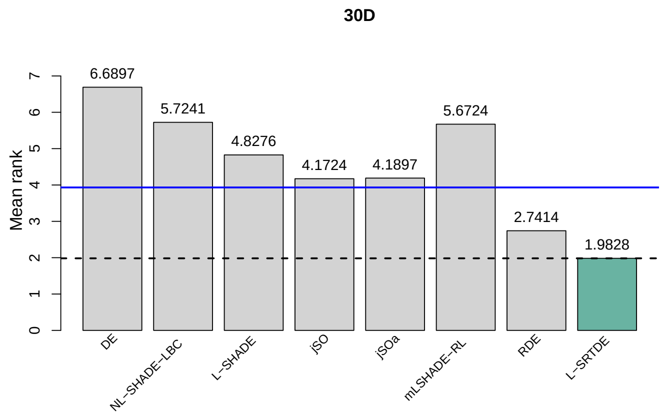

Figure 2. Friedman ranks of algorithms with a critical distance for 30D.

In Figures 1–4, a critical distance indicates that there exists a statistical difference among the two compared algorithms if the difference of their Friedman rank values is smaller than the critical value. We plot a dotted line of the best-obtained algorithm and the blue line to show the critical distance compared to the best-performed algorithm, which is depicted in green.

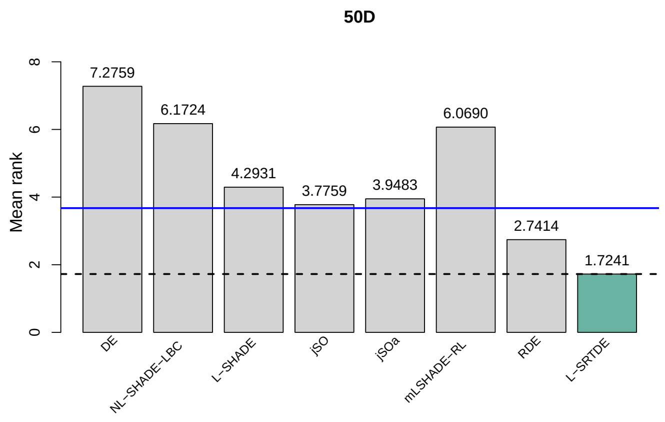

Figure 3. Friedman ranks of algorithms with a critical distance for 50D.

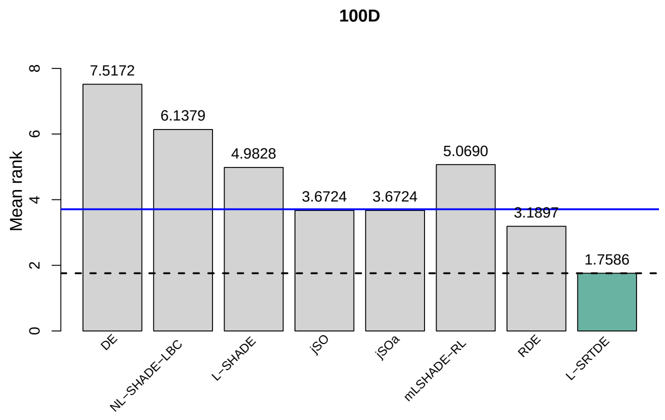

Figure 4. Friedman ranks of algorithms with a critical distance for 100D.

Figure 1 shows the comparison for $D = 10$, and we can see that all eight algorithms are below the critical distance from L-SRTDE, which is the best algorithm. This means there are no significant differences between all eight algorithms. Table 12 tabulates Friedman ranks, shows a performance order in column Rank, and at the bottom of the table, the χ^2 and p values of the Friedman test are reported. If the p-value is lower than predefined significance level (α) it indicates that the algorithms have statistical significant differences in performance.

The obtained results for $D = 30$, $D = 50$, and $D = 100$ are shown in Figures 2–4, respectively. One can see that L-SRTDE is statistically better than DE, L-SHADE, NL-SHADE-LBC, jSO, jSOa, and mLSHADE-RL, while there is no statistical difference between L-SRTDE and RDE. This holds for each dimension of $D = 30$, $D = 50$, and $D = 100$.

On $D = 100$, we can also see that algorithms RDE, jSOa, and jSO are close to the blue line (i.e., critical distance), and RDE is the only one that is slightly below the blue line.

In general, all three tests that were used in the comparison of eight algorithms report very consistent comparison results for each dimension.

7. Discussion

The classic DE algorithm lacks the sophisticated mechanisms introduced in the algorithms, which were proposed later. While it is generally effective, it struggles with maintaining diversity and balancing exploration and exploitation without additional enhancements. It does not have adaptive mechanisms for F and Cr, leading to suboptimal parameter tuning. It does not have an external archive or hybrid strategies, limiting diversity control. It does not use a restart mechanism, making it prone to premature convergence.

The ranking of compared algorithms reflects each algorithm's ability to balance exploration and exploitation, maintain population diversity, and adapt to different stages of evolution. L-SRTDE leads due to its success rate-driven adaptation, refining both mutation and scaling. The repaired crossover rate (Cr_a) refines the crossover mechanism, ensuring more effective variation across dimensions. RDE follows closely, leveraging hybrid strategies and adaptive resource allocation. It combine "DE/current-to-pbest/1" and "DE/current-to-order-pbest/1". The key improvement is in how resources are allocated between these strategies—an adaptive mechanism evaluates their average fitness improvement and adjusts accordingly. Additionally, RDE incorporates rank pressure-based selection, refining the probability of selecting individuals based on fitness ranks. jSOa stands out for its archive management, ensuring stability. Instead of randomly deleting older solutions, jSOa divides the archive into "better" and "worse" sections based on functional values. This ensures that stronger solutions are preserved while allowing some weaker solutions to be refreshed. This more structured archive update leads to better diversity retention without compromising convergence speed. jSO remains strong, though jSOa refines its archive. mLSHADE-RL's multi-operator approach seems to be powerful. Non-linear population size reduction strategy combined with linear bias change for both the scaling factor F and the crossover rate Cr improve efficiency of the NL-SHADE-LBC algorithm. The L-SHADE algorithm, with its robust adaptive mechanism and dynamic population control, serves as the basis for many enhancements to the DE algorithm.

Each of these algorithms offers unique strengths, but L-SRTDE currently represents the most advanced evolutionary strategy in this group of algorithms.

8. Conclusions

In this study, we conducted a comprehensive review of state-of-the-art algorithms based on Differential Evolution (DE) that have been proposed in recent years, including L-SHADE, jSO, jSOa, NL-SHADE-LBC, mLSHADE-RL, RDE, and L-SRTDE, along with classic DE. We examined the key mechanisms incorporated into these algorithms and systematically compared their performance. Specifically, we identified shared and unique mechanisms among the algorithms.

To ensure robust performance evaluation, we employed statistical comparisons, which provide reliable insights into algorithm efficacy. The Wilcoxon signed-rank test was utilized for pairwise comparisons, while the Friedman test was applied for multiple comparisons. Additionally, the Mann-Whitney U-score test was incorporated to enhance the statistical rigor of our analysis. We also performed a cumulative assessment of the eight algorithms and evaluated their performance across different function families, including unimodal, multimodal, hybrid, and composition functions.

The experimental evaluation was conducted on benchmark problems defined for the CEC'24 Special Session and Competition on Single Objective Real Parameter Numerical Optimization. We analyzed problem dimensions of 10, 30, 50, and 100, running all algorithms independently with parameter settings as recommended by the original authors.

The key findings from our experiments indicate that all three statistical tests consistently ranked the algorithms in the following order: L-SRTDE achieved the highest rank, followed by RDE. Next are jSOa and jSO, which alternate between third and fourth positions. LSHADE consistently follows in fifth, succeeded by mLSHADE-RL and NL-SHADE-LBC, with DE ranking last. Notably, a similar ranking was obtained by the method proposed in [8], which was also employed for algorithm ranking in CEC 2024 [34]. However, CEC 2024 competition was performed only at dimension $D = 30$ [34]. We extend the analysis to a lower dimension ($D = 10$) and higher dimensions ($D = 50$ and $D = 100$). The analysis across different function families ranks the algorithms slightly differently at dimension $D = 10$; however, at $D = 100$, L-SRTDE remains the best across all function families.

In conclusion, significant progress has been made since the first published Differential Evolution algorithm. The enhanced versions of DE now incorporate a variety of mechanisms, all of which collectively contribute to improved performance.

Author Contributions: Conceptualization, M.S.M.; methodology, M.S.M. and J.B.; software, J.B.; validation, M.S.M. and J.B.; investigation, M.S.M. and J.B.; writing—original draft preparation, J.B. and M.S.M.; writing—review and editing, J.B. and M.S.M.; visualization, J.B. Both authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Slovenian Research and Innovation Agency (research core funding No. P2-0069 — Advanced Methods of Interaction in Telecommunications and P2-0041 — Computer Systems, Methodologies, and Intelligent Services).

Data Availability Statement: The data is available as supplementary material.

Acknowledgments: The authors would like to thank the developers of the jSOa, mLSHADE-RL, RDE, and L-SRTDE algorithms for making their source code publicly available.

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

References

1. Storn, R.; Price, K. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization* **1997**, *11*, 341–359.
2. Price, K. *Differential Evolution: a Practical Approach to Global Optimization*; Springer Science & Business Media, 2005.
3. Das, S.; Mullick, S.S.; Suganthan, P.N. Recent advances in differential evolution–an updated survey. *Swarm and evolutionary computation* **2016**, *27*, 1–30.
4. Opara, K.R.; Arabas, J. Differential Evolution: A survey of theoretical analyses. *Swarm and evolutionary computation* **2019**, *44*, 546–558.
5. Pant, M.; Zaheer, H.; Garcia-Hernandez, L.; Abraham, A.; et al. Differential Evolution: A review of more than two decades of research. *Engineering Applications of Artificial Intelligence* **2020**, *90*, 103479.
6. Derrac, J.; García, S.; Molina, D.; Herrera, F. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation* **2011**, *1*, 3–18.
7. Carrasco, J.; García, S.; Rueda, M.; Das, S.; Herrera, F. Recent trends in the use of statistical tests for comparing swarm and evolutionary computing algorithms: Practical guidelines and a critical review. *Swarm and Evolutionary Computation* **2020**, *54*, 100665.
8. Price, K.V.; Kumar, A.; Suganthan, P.N. Trial-based dominance for comparing both the speed and accuracy of stochastic optimizers with standard non-parametric tests. *Swarm and Evolutionary Computation* **2023**, *78*, 101287.
9. Neri, F.; Tirronen, V. Recent advances in differential evolution: a survey and experimental analysis. *Artificial intelligence review* **2010**, *33*, 61–106.
10. Das, S.; Suganthan, P.N. Differential evolution: A survey of the state-of-the-art. *IEEE transactions on evolutionary computation* **2011**, *15*, 4–31.
11. Dragoi, E.N.; Dafinescu, V. Parameter control and hybridization techniques in differential evolution: a survey. *Artificial Intelligence Review* **2016**, *45*, 447–470.
12. Tanabe, R.; Fukunaga, A. Reviewing and benchmarking parameter control methods in differential evolution. *IEEE Transactions on Cybernetics* **2019**, *50*, 1170–1184.
13. Al-Dabbagh, R.D.; Neri, F.; Idris, N.; Baba, M.S. Algorithmic design issues in adaptive differential evolution schemes: Review and taxonomy. *Swarm and Evolutionary Computation* **2018**, *43*, 284–311.
14. Reyes-Davila, E.; Haro, E.H.; Casas-Ordaz, A.; Oliva, D.; Avalos, O. Differential Evolution: A Survey on Their Operators and Variants. *Archives of Computational Methods in Engineering* **2024**, pp. 1–30.
15. Piotrowski, A.P. Review of differential evolution population size. *Swarm and Evolutionary Computation* **2017**, *32*, 1–24.
16. Parouha, R.P.; Verma, P. State-of-the-art reviews of meta-heuristic algorithms with their novel proposal for unconstrained optimization and applications. *Archives of Computational Methods in Engineering* **2021**, *28*, 4049–4115.
17. Ma, Z.; Wu, G.; Suganthan, P.N.; Song, A.; Luo, Q. Performance assessment and exhaustive listing of 500+ nature-inspired metaheuristic algorithms. *Swarm and Evolutionary Computation* **2023**, *77*, 101248.
18. Goula, E.M.K.; Sotiropoulos, D.G. HRA: A Multi-Criteria Framework for Ranking Metaheuristic Optimization Algorithms. *arXiv preprint arXiv:2409.11617* **2024**.

19. Piotrowski, A.P.; Napiorkowski, J.J.; Piotrowska, A.E. Choice of benchmark optimization problems does matter. *Swarm and Evolutionary Computation* **2023**, *83*, 101378.
20. Tanabe, R.; Fukunaga, A.S. Improving the search performance of SHADE using linear population size reduction. In Proceedings of the 2014 IEEE congress on evolutionary computation (CEC). IEEE, 2014, pp. 1658–1665.
21. Tanabe, R.; Fukunaga, A. Evaluating the performance of SHADE on CEC 2013 benchmark problems. In Proceedings of the 2013 IEEE Congress on evolutionary computation. IEEE, 2013, pp. 1952–1959.
22. 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). IEEE, 2022, pp. 01–08.
23. 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). IEEE, 2021, pp. 809–816.
24. Brest, J.; Maučec, M.S.; Bošković, B. Single objective real-parameter optimization: Algorithm jSO. In Proceedings of the 2017 IEEE congress on evolutionary computation (CEC). IEEE, 2017, pp. 1311–1318.
25. Brest, J.; Maučec, M.S.; Bošković, B. iL-SHADE: Improved L-SHADE algorithm for single objective real-parameter optimization. In Proceedings of the 2016 IEEE Congress on Evolutionary Computation (CEC). IEEE, 2016, pp. 1188–1195.
26. Bujok, P. Progressive Archive in Adaptive jSO Algorithm. *Mathematics* **2024**, *12*, 2534.
27. Chauhan, D.; Trivedi, A.; et al. A Multi-operator Ensemble LSHADE with Restart and Local Search Mechanisms for Single-objective Optimization. *arXiv preprint arXiv:2409.15994* **2024**.
28. Awad, N.H.; Ali, M.Z.; Suganthan, P.N.; Reynolds, R.G. An ensemble sinusoidal parameter adaptation incorporated with L-SHADE for solving CEC2014 benchmark problems. In Proceedings of the 2016 IEEE congress on evolutionary computation (CEC). IEEE, 2016, pp. 2958–2965.
29. Awad, N.H.; Ali, M.Z.; Suganthan, P.N. Ensemble sinusoidal differential covariance matrix adaptation with Euclidean neighborhood for solving CEC2017 benchmark problems. In Proceedings of the 2017 IEEE congress on evolutionary computation (CEC). IEEE, 2017, pp. 372–379.
30. Tao, S.; Zhao, R.; Wang, K.; Gao, S. An Efficient Reconstructed Differential Evolution Variant by Some of the Current State-of-the-art Strategies for Solving Single Objective Bound Constrained Problems. *arXiv preprint arXiv:2404.16280* **2024**.
31. 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). IEEE, 2018, pp. 1–8.
32. Stanovov, V.; Semenkin, E. Success Rate-based Adaptive Differential Evolution L-SRTDE for CEC 2024 Competition. In Proceedings of the 2024 IEEE Congress on Evolutionary Computation (CEC). IEEE, 2024, pp. 1–8.
33. Stanovov, V.; Akhmedova, S.; Semenkin, E. Dual-Population Adaptive Differential Evolution Algorithm L-NTADE. *Mathematics* **2022**, *10*, 4666.
34. Qiao, K.; Ban, X.; Chen, P.; Price, K.V.; Suganthan, P.N.; Wen, X.; Liang, J.; Wu, G.; Yue, C. Performance comparison of CEC 2024 competition entries on numerical optimization considering accuracy and speed. PPT Presentation, 2000.
35. Qiao, K.; Wen, X.; Ban, X.; Chen, P.; Price, K.; Suganthan, P.; Liang, J.; Wu, G.; Yue, C. Evaluation criteria for CEC 2024 competition and special session on numerical optimization considering accuracy and speed. *Zhengzhou University, Central South University, Henan Institute of Technology, Qatar University, Tech. Rep.* **2023**.