symmetry

Article

NLAPSMjSO-EDA: A Nonlinear Shrinking Population Strategy Algorithm for Elite Group Exploration with Symmetry Applications

Yong Shen , Jiaxuan Liang, Hongwei Kang *, Xingping Sun and Qingyi Chen

School of Software, Yunnan University, Kunming 650000, China; sheny@ynu.edu.cn (Y.S.); liangjiaxuan@mail.ynu.edu.cn (J.L.); sunxp@ynu.edu.cn (X.S.); devas9@ynu.edu.cn (Q.C.)
* Correspondence: hwkang@ynu.edu.cn

Abstract: This work effectively modifies APSM-jSO (a novel jSO variant with an adaptive parameter selection mechanism and a new external archive updating mechanism) to offer a new jSO (single objective real-parameter optimization: Algorithm jSO) version called NLAPSMjSO-EDA. There are three main distinctions between NLAPSMjSO-EDA and APSM-jSO. Firstly, in the linear population reduction strategy, the number of individuals eliminated in each generation is insufficient. This results in a higher number of inferior individuals remaining, and since the total number of iterations is fixed, these inferior individuals will also consume iteration counts for their evolution. Therefore, it is essential to allocate more iterations to the elite population to promote the emergence of superior individuals. The nonlinear population reduction strategy effectively addresses this issue. Secondly, we have introduced an Estimation of Distribution Algorithm (EDA) to sample and generate individuals from the elite population, aiming to produce higher-quality individuals that can drive the iterative evolution of the population. Furthermore, to enhance algorithmic diversity, we increased the number of individuals in the initial population during subsequent experiments to ensure a diverse early population while maintaining a constant total number of iterations. Symmetry plays an essential role in the design and performance of NLAPSMjSO-EDA. The nonlinear population reduction strategy inherently introduces a form of asymmetry that mimics natural evolutionary processes, favoring elite individuals while reducing the influence of inferior ones. This asymmetric yet balanced approach ensures a dynamic equilibrium between exploration and exploitation, aligning with the principles of symmetry and asymmetry in optimization. Additionally, the incorporation of EDA utilizes probabilistic symmetry in sampling from the elite population, maintaining structural coherence while promoting diversity. Such applications of symmetry in algorithm design not only improve performance but also provide insights into balancing diverse algorithmic components. NLAPSMjSO-EDA, evaluated on the CEC 2017 benchmark suite, significantly outperforms recent differential evolution algorithms. In conclusion, NLAPSMjSO-EDA effectively enhances the overall performance of APSM-jSO, establishing itself as an outstanding variant combining jSO and EDA algorithms. The algorithm code has been open-sourced.

Keywords: non-linear population shrinking; symmetry; estimation of distribution algorithm; single objective real-parameter optimization; evolutionary computation

check for updates

Academic Editor: Sergei Odintsov

Received: 13 November 2024
Revised: 6 January 2025
Accepted: 15 January 2025
Published: 21 January 2025

Citation: Shen, Y.; Liang, J.; Kang, H.; Sun, X.; Chen, Q. NLAPSMjSO-EDA: A Nonlinear Shrinking Population Strategy Algorithm for Elite Group Exploration with Symmetry Applications. *Symmetry* **2025**, *17*, 153. https://doi.org/10.3390/sym17020153

1. Introduction

The optimization problem spans multiple fields such as mathematics, engineering, computer science, economics, physics, biology, and operations research [1]. Optimization is ubiquitous in daily life and is applicable across various practical problems that can be formulated using optimization models. It is a wide field of study that suggests certain approaches to solve certain types of problems, including non-convex optimization, quadratic programming, integer programming, and linear programming. In order to address a wide range of issues, researchers concentrate on creating generic algorithms, which are further divided into single- and multi-objective algorithms according to their goals [2,3]. Problems are also classified as constrained [4] or unconstrained, each playing increasingly significant roles in practical engineering applications [5,6].

Unconstrained single-objective optimization is a basic kind of optimization problem that serves as the foundation for more complicated optimization issues. It finds extensive applications in economic scheduling [7,8], production planning [9], engineering optimization problems [10,11], image processing [12], and other domains [13]. Solving unconstrained single-objective real-parameter optimization problems remains a challenging task.

The Differential Evolution (DE) method was first presented by Price and Storn [14], and is favored for its practical implementation, robustness, and excellent performance in single-objective optimization problems. During iterations, it employs mutation, crossover, and selection operations to guide individuals in the population toward global optima. However, DE's performance heavily relies on mutation strategies and control parameters [15,16]. The proper selection of these influences significantly enhances population convergence speed and diversity. Conversely, poor choices can lead to decreased performance, causing premature convergence or stagnation [17,18], limiting exploration and exploitation in subsequent generations. Therefore, ongoing research focuses on optimizing DE.

In the population-based optimization technique known as DE, individuals develop within a population and the optimization results are influenced by their quality. Throughout optimization iterations, individuals may gain valuable evolutionary directions and information, but they may also acquire misleading cues without adjustment mechanisms, potentially yielding adverse optimization results. Thus, maintaining good population diversity is crucial. New evolutionary directions and information are acquired through mutations. Mutation strategies determine search baselines, and new differential vectors dictate search directions. To create trial vectors, crossover mixes mutation vectors—which are produced via mutation—with target vectors, which are members of the population. After trial vector fitness values are assessed using assessment criteria, valuable individuals keep evolving.

In DE algorithms, different mutation strategies exhibit varying search performances [19,20]. Strategies like DE/rand/1 and DE/rand/2 initiate searches from randomly selected individuals, performing well in global exploration. DE/best/1 and DE/best/2 initiate searches from the best individual. DE/current-to-pbest/1 linearly combines the current individual with a randomly selected p% of high-fitness individuals as starting points, balancing global and local searches. Combining these strategies can enhance optimization performance during the search for optimal solutions [21,22].

Important control elements in DE include population size NP, crossover rate CR, and scaling factor F. Among them, NP receives relatively less attention but low NP may result in the loss of valuable individual information, hindering population convergence to global optima. Currently, no definitive NP size has been proven to be superior, but effective solutions like the L-shade algorithm series [23,24] dynamically adjust population size to avoid unnecessary computations. During optimization, there will inevitably be a loss of population variety and a slowdown in the rate of convergence. Excessive adherence to global exploration strategies increases diversity but may slow convergence, making it challenging to find optimal individuals. Conversely, excessive convergence within the population decreases diversity, potentially trapping solutions in local optima.

To address these challenges, this study improves upon existing approaches by integrating the latest variant of jSO, APSM-jSO, modifying population reduction strategies to allow more generations for evolution, and introducing EDA algorithms to enhance population diversity. Changes in population reduction strategies enable more generations to explore optimal solutions, enhancing local exploration capabilities. Additionally, integrating EDA algorithms alleviates the dilemma of late-stage diversity loss in jSO algorithms, indirectly boosting global exploration capabilities.

2. Related Work

The "DE/current-to-pbest" mutation approach was first presented by Jade [25]. It is still recognized as the foundation for new variations based on DE and uses representative solutions kept in an external archive, together with the most representative adaptive parameter management strategy.

By incorporating concepts from iL-SHADE [23], jSO [2], which was placed second in the IEEE CEC 2017 competition, presented the DE/current-to-pbest-w/1 weighted mutation approach in 2017. In the early and late phases of optimization, jSO uses varying scaling factors to balance exploration and exploitation capabilities. Also, in the same year Noor H. Awad et al. [26] adjusted scaling factors using two sine waves: non-adaptive sine decrease and adaptive sine increase. This approach employed a performance-adaptive mechanism based on early successes rather than random selection of sine waves. Additionally, the crossover operator utilized Euclidean neighborhood covariance matrix learning to establish suitable coordinates, enhancing LSHADE-Epsin's ability to handle highly correlated variables (The introduction of various algorithms is shown in Table 1).

Table 1. Introduction of Algorithms.

Algorithm	Year Proposed	Description
DE	1997	Standard Differential Evolution (DE) is an evolutionary algorithm for global optimization, progressively improving candidate solutions in the population through differential mutation, crossover, and selection operations to find optimal solutions for complex problems [14].
L-SHADE	2014	This algorithm is an improvement upon SHADE [27], adjusting population size through Linear Population Size Reduction (LPSR) during evolution to optimize population scale [23].
L-SHADE-Epsin	2017	Enhanced version of L-SHADE integrating performance-adaptive sine methods and covariance matrix learning in crossover operators [26].
EBOwithCMAR	2017	Covariance Matrix Adaptation with Retrospective (CMAR) improves local search capabilities of Evolutionary Bridge Optimization (EBO) by generating new solutions using the covariance matrix. This variant is referred to as EBOwithCMAR [28].
MadDE	2021	A variant of LSHADE that makes use of several mutation techniques to take advantage of different adaptation strategies [29].
IDE-EDA	2023	Algorithm combining Differential Evolution Algorithm (LSHADE-Rsp) with Estimation of Distribution Algorithm (EDA) [30].
APSM-jSO	2023	New variant of jSO proposed by effective modifications, incorporating strategies from LSHADE-RSP [31].

Abhishek Kumar introduced Covariance Matrix Adaptation with Retrospective (CMAR), which generates new solutions using the covariance matrix, thereby enhancing local search capabilities for EBO. This version of EBO is referred to as EBOwithCMAR [28], winning the IEEE CEC 2017 competition.

Stanovov et al.'s LSHADE-RSP [32], which employed a rank-based selection pressure mechanism in lieu of randomly picked people in the mutation strategy was placed second in the IEEE CEC 2018 competition.

Xia et al. [33] presented a unique fitness-driven hybrid in 2021 that takes into account both fitness and novelty in order to balance exploration and exploitation capabilities. They proposed the new DE-based version NFDDE.

Stanovov et al. [34] introduced the NL-SHADE-RSP method in 2021. It makes use of non-linear population reduction, crossover rate control, and adaptive archive use.

Yintong Li proposed IDE-EDA in 2023 [30], which combines the Estimation of Distribution Algorithm (EDA) with the Differential Evolution Algorithm (LSHADE-Rsp).

Also, in 2023 a new variant of jSO, APSM-jSO, was proposed by applying effective modifications to jSO, utilizing strategies from LSHADE-RSP [31].

The explanation of various abbreviations is shown in Table 2 .

Table 2. Abbreviations and Terms of DE Algorithms.

Abbreviation	Full Name	Description
DE	Differential Evolution	A standard evolutionary algorithm for global optimization, improving candidate solutions using mutation, crossover, and selection.
SHADE	Success-History Based Adaptive Differential Evolution	A variant of differential evolution that adapts the scaling factor and crossover rate based on the success history of previous generations. It improves the balance between exploration and exploitation during optimization.
L-SHADE	Linear Population Size Reduction SHADE	An improvement of SHADE that adjusts population size dynamically using LPSR during optimization.
L-SHADE-Epsin	Enhanced Performance SHADE with Covariance Matrix	A version of L-SHADE integrating adaptive sine methods and covariance matrix learning for crossover operations.
EBOwithCMAR	Evolutionary Bridge Optimization with Covariance Matrix Adaptation and Retrospective	Combines covariance matrix learning with EBO to enhance local search capabilities.
MadDE	Multi-Adaptive Differential Evolution	A variant of L-SHADE that applies multiple adaptation techniques for improved optimization.
IDE-EDA	Improved Differential Evolution with Estimation of Distribution Algorithm	Combines L-SHADE with EDA to generate high-quality solutions in high-dimensional spaces.
APSM-jSO	Adaptive Parameter Selection Mechanism-jSO	A jSO variant integrating L-SHADE-RSP strategies for effective parameter selection.
EDA	Estimation of Distribution Algorithm	A population-based optimization algorithm that models the distribution of the best candidate solutions and uses it to generate new solutions. It integrates probabilistic models to guide the search process.

2.1. Differential Evolution

A uniform distribution is used in the DE standard technique to produce the starting population P [14]. Equation (1) illustrates how each potential solution x_i^j is randomly started at the beginning of the evolution process.

$$x_i^j = x_{\min}^j + \text{rand} \cdot (x_{\max}^j - x_{\min}^j), \quad i \in \{1,2,\ldots,NP\}, \quad j \in \{1,2,\ldots,D\} \tag{1}$$

Here, $x_{\min}^j$ and $x_{\max}^j$ represent the problem's lower and upper bounds in dimension j; NP represents the population size, D represents the dimension size, and rand is a uniformly distributed number between 0 and 1.

During the evolution process, it performs mutation, selection, and crossover operations. The five widely used classic mutation strategies in DE, namely DE/rand/1, DE/rand/2, DE/best/1, DE/current-to-best/1, and DE/rand-to-best/1, are described by Equations (2)–(6).

$$v_i = x_{r_1} + F \cdot (x_{r_2} - x_{r_3}) \tag{2}$$

$$v_i = x_{r_1} + F \cdot (x_{r_2} - x_{r_3}) + F \cdot (x_{r_4} - x_{r_5}) \tag{3}$$

$$v_i = x_{best} + F \cdot (x_{r_1} - x_{r_2}) \tag{4}$$

$$v_i = x_i + F \cdot (x_{best} - x_i) + F \cdot (x_{r_1} - x_{r_2}) \tag{5}$$

$$v_i = x_{r_1} + F \cdot (x_{best} - x_{r_1}) + F \cdot (x_{r_2} - x_{r_3}) \tag{6}$$

where $r_1, r_2, r_3, r_4, r_5 \in \{1,2,3,\ldots,NP\}$ are uniformly distributed random indices that are mutually exclusive. F is the scaling factor, x_{best} is the currently best individual obtained. When v_i exceeds the lower or upper bounds, it is corrected using the strategy shown in Equation (7).

$$v_i^j = \begin{cases} \left(L^j + x_i^j\right)/2, & v_i^j < L_j \\ \left(U^j + x_i^j\right)/2, & v_i^j > U_j \end{cases} \tag{7}$$

Using binomial crossover operation, the trial vector u_i is generated based on the mutation vector v_i, as shown in Equation (8).

$$u_i^j = \begin{cases} v_i^j, & \text{if rand} < Cr \text{ or } j = j_{\text{rand}} \\ x_i^j, & \text{otherwise} \end{cases} \tag{8}$$

where $j_{\text{rand}} \in \{1,2,3,\ldots,D\}$ is a uniformly random index, and Cr is the crossover rate.

Equation (9) illustrates how the population is updated using a greedy method based on the following fitness function:

$$\boldsymbol{x}_i = \begin{cases} \boldsymbol{u}_i, f(\boldsymbol{u}_i) \le f(\boldsymbol{x}_i) \\ \boldsymbol{x}_i, \text{otherwise} \end{cases} \tag{9}$$

where $f(\cdot)$ denotes the specific problem's fitness function.

2.2. LSHADE

L-SHADE [23] algorithm enhances algorithm performance by introducing a linearly decreasing population size in the SHADE algorithm, focusing on both exploration and exploitation. The population reduction formula is shown in Equation (10):

$$NP = \text{round}\left(\frac{FES}{FES_{\max}}(NP_{\min} - NP_{\text{init}})\right) + NP_{\text{init}} \tag{10}$$

The rounding procedure is indicated by round in the formula above, and the maximum and current iteration counts are represented by $FES_{\max}$ and FES, respectively. The initial and minimal population sizes are denoted by NP_{init} and $NP_{\min}$, respectively.

In Equation (13), $M_{CR,k}$ represents the k-th element in the historical memory used to update the control parameter CR. This formula dynamically adjusts the value of CR as follows: If $M_{CR,k}$ has not been initialized or if the number of successful individuals Scr in the current generation is zero, $M_{CR,k}$ remains in an undefined state $\perp$. Otherwise, it is updated using the weighted Lehmer mean $\text{mean}_{\text{WL}}(\mathcal{S}_{\text{CR}})$ of the successful CR values. The weighted Lehmer mean is calculated as $\text{mean}_{\text{WL}}(\mathcal{S}_{\text{CR}}) = \frac{\sum_n w_n \cdot s_n^2}{\sum_n w_n \cdot s_n}$, where w_n denotes the weight (in Equation (12)), typically associated with fitness improvement, and s_n represents elements in the set of successful CR values. This mechanism ensures that the update of $M_{CR,k}$ adapts dynamically to the optimization performance in each generation, thereby enhancing the algorithm's search efficiency and robustness.

$$\text{mean}_{WL}(S) = \frac{\sum_{n=1}^{|S|} \omega_n \cdot S_n^2}{\sum_{n=1}^{|S|} \omega_n \cdot S_n} \tag{11}$$

$$\omega_n = \frac{|f(\boldsymbol{u}_n) - f(\boldsymbol{x}_n)|}{\sum_{n=1}^{|S|} |f(\boldsymbol{u}_n) - f(\boldsymbol{x}_n)|} \tag{12}$$

$$M_{CR,k} = \begin{cases} \perp, & M_{CR,k} = \perp \ \text{ or } \ \max(\text{Scr} = 0) \\ \text{mean}_{\text{WL}}(\mathcal{S}_{\text{CR}}), & \text{otherwise} \end{cases} \tag{13}$$

Equation (14) defines the control parameter Cr_i. If $M_{CR,k} = \perp$, indicating that the historical memory is undefined or invalid, Cr_i is set to 0. Otherwise, Cr_i is generated randomly around the center $\text{M}_{\text{CR,R}}$ with a deviation of 0.1, as defined by $\text{rand}(\text{M}_{\text{CR,R}}, 0.1)$. This approach ensures that Cr_i remains stable when the memory is invalid while introducing randomness for exploration when the memory is valid.

$$Cr_i = \begin{cases} 0, & M_{CR,k} = \perp \\ \text{rand}(\text{M}_{\text{CR,R}}, 0.1), & \text{otherwise} \end{cases} \tag{14}$$

In the L-SHADE algorithm, the greediness factor p is typically set to 0.2, which randomly selects the best individuals from the top 20% elite, with the archive size set to $NP \times Arate$, where $Arate$ controls the size of the external archive.

2.3. LSHADE-Epsin

In the first phase of LSHADE-Epsin, the mutation factor F for each target vector in generation g is defined as shown in Equation (15):

$$F_{i,g} = \frac{1}{2}\left(\sin(2\pi \cdot \text{freq} \cdot g + \pi) \cdot \frac{G_{\max} - g}{G_{\max}} + 1\right) \tag{15}$$

Equation (15) defines the parameter $F_{i,g}$, which dynamically adjusts based on the generation index g. The formula includes a sinusoidal component modulated by the frequency freq and the current generation g, combined with a linear decay factor. Specifically:

- The term $\sin(2\pi \cdot \text{freq} \cdot g + \pi)$ introduces oscillations in $F_{i,g}$ over generations, with the frequency controlled by freq.
- The factor $\frac{G_{\max} - g}{G_{\max}}$ ensures a gradual decrease in $F_{i,g}$ as the generation g approaches the maximum generation $G_{\max}$, simulating a decay effect.

- The addition of 1 and division by 2 normalize the oscillation range, ensuring that $F_{i,g}$ remains within $[0, 1]$.

This design allows $F_{i,g}$ to balance exploration and exploitation over generations, leveraging oscillations for diversity and decay for convergence.

Here, g represents the current generation, freq is a fixed value (freq = 0.5), i represents the index of the individual, i ranges from 1 to the population size NP, and $G_{\max}$ is the maximum number of iterations.

As demonstrated by Equation (16), an adaptive sine rise adjustment is used in the second arrangement.

$$F_{i,g} = \frac{1}{2}\left(\sin(2\pi \cdot \mathrm{freq}_i \cdot g) \cdot \frac{g}{G_{\max}} + 1\right) \tag{16}$$

In Equation (17), $\mathrm{freq}_{i,g}$ represents the frequency for the i-th individual at generation g. It is generated using the function $\mathrm{randc}(\mathrm{ufreq}_{ri}, 0.1)$, where randc introduces controlled randomness.

The term ufreq_{ri} serves as the central value or the mean around which the randomness is applied. The value 0.1 specifies the range or deviation for the random sampling, ensuring that the generated frequency remains close to ufreq_{ri}.

The randc function is designed to create a bounded random value while maintaining a controlled level of variability, supporting diversity among individuals in the population without deviating excessively from the baseline value ufreq_{ri}.

$$\mathrm{freq}_{i,g} = \mathrm{randc}(\mathrm{ufreq}_{ri}, 0.1) \tag{17}$$

The average success frequencies from prior generations are stored in an external memory freqM, from which ufreq_{ri} is chosen at random. A random index r_1 is selected from $[1, h]$ at the conclusion of generation g.

In the first phase, F is chosen to increase or decrease based on learning records' probabilities.

In the second phase of LSHADE-Epsin, except for F using a Cauchy distribution, the algorithm is similar to LSHADE.

Furthermore, for every algorithm generation, the crossover operator learns a covariance matrix with probability p_c based on Euclidean neighborhoods. The best person is first determined by sorting the people according to their fitness ratings BestX. Then, we compute the Euclidean distances between BestX and every other person in the population. Individuals are classified, creating communities around the best individuals based on their distances and NP. The percentage of people utilized to create the covariance matrix C is p_s. $NP \times p_s$ also drops dynamically as a result of the population size dynamically shrinking. This neighborhood is used to construct the covariance matrix C, as indicated by Equations (18)–(20):

$$C = BDB^T \tag{18}$$

In this case, D is a diagonal matrix made up of eigenvalues, while B^T and B are orthogonal matrices.

Next, B^T is used to update the target vectors $X'_{i,g}$ and trial vectors $V'_{i,g}$, as shown in Equations (19) and (20):

$$X'_{i,g} = B^T X'_{i,g} \tag{19}$$

$$V'_{i,g} = B^T V'_{i,g} \tag{20}$$

Equation (21) defines the trial vector $\mathbf{u}'_{i,j,g}$ for the i-th individual at dimension j and generation g. This formula is a key part of the mutation and crossover process in differential evolution. Specifically,

- Case 1: If $\text{rand}(0,1) \leq CR_{i,g}$ or $j = j_{\text{rand}}$, the component $V'_{i,j,g}$ from the donor vector is selected. $\text{rand}(0,1)$ is a random number uniformly distributed between 0 and 1. $CR_{i,g}$ is the crossover rate for individual i at generation g, controlling the probability of inheriting components from the donor vector. j_{rand} ensures that at least one component of the trial vector $\mathbf{u}'_{i,j,g}$ is inherited from the donor vector, preventing the trial vector from being identical to the target vector.
- Case 2: Otherwise, the component $X'_{i,j,g}$ from the target vector is retained.

This mechanism balances exploration and exploitation by probabilistically mixing components from the donor vector and the target vector, encouraging diversity in the population while preserving some characteristics of the original individuals.

$$\mathbf{U}'_{i,j,g} = \begin{cases} V'_{i,j,g}, & \text{if } \text{rand}(0,1) \leq CR_{i,g} \text{ or } j = j_{\text{rand}} \\ X'_{i,j,g}, & \text{otherwise} \end{cases} \tag{21}$$

Lastly, as shown by Equation (22), a binomial crossover is performed with $X'_{i,g}$ and $V'_{i,g}$ to produce a trial vector $U'_{i,g}$ in the eigenspace.

$$U_{i,g} = B \cdot U'_{i,g} \tag{22}$$

The vectors are subsequently converted back to the initial coordinate system.

2.4. EBOwithCMAR

The framework of EBOwithCMAR draws inspiration from the algorithm proposed in UMOEAs-II [35]. The EBOwithCMAR algorithm begins by randomly generating an initial population PS within the search boundary (initial solutions). Three sub-populations of sizes PS_1, PS_2, and PS_3 are then created from this population. While the third population is used for CMAR, the first and second populations are used for EBO processing. During the evolution process, the use of the EBO or CMAR algorithm is determined by probabilities $prob_1$ and $prob_2$. These probabilities are not fixed and are updated as described later. In this algorithm, a cycle is referred to as a control step (CS). The values of $prob_1$ and $prob_2$ are initialized to 1 at the beginning of each CS.

(1) Superiority of Optimal Solutions Found by EBO and CMAR: This step evaluates the quality of solutions produced by EBO (Enhanced Biogeography-based Optimization) and CMAR (Cultural Memory Adaptive Rotation). EBO is responsible for the efficient exploration of the search space through migration and mutation, while CMAR focuses on enhancing exploitation by refining solutions using adaptive rotational strategies informed by cultural memory. The combined approach leverages the strengths of both techniques, and the superiority of the resulting solutions is demonstrated by comparing their objective function values to those of standalone methods or benchmarks, showcasing an effective balance of exploration and exploitation.

(2) Diversity Ratio of PS1 and PS3: This step measures the diversity ratio between two solution subsets, PS1 and PS3, to evaluate the algorithm's ability to maintain diversity during optimization. PS1 typically represents diverse candidate solutions emphasizing exploration, while PS3 focuses on refined solutions in promising regions for exploitation. The diversity ratio ensures that the algorithm avoids premature convergence or stagnation, effectively adapting to different stages of the optimization process. This metric reflects the dynamic balance achieved by EBOwithCMAR.

At the end of each CS, data sharing occurs, and the values of $prob_1$ and $prob_2$ are reset to 1 [35]. The two algorithms then proceed to the next cycle, repeating the same process. In order to optimize the developmental potential of EBOwithCMAR in the final phases of the method, sequential quadratic programming (SQP) is utilized with the dynamic probability $prob_{ls}$ when 75% of the process has been finished. In the event that the solution discovered by local exploration with SQP is superior to the existing solutions, $prob_{ls}$ will be set to its starting value. Should this not be the case, $prob_{ls}$ is assigned a very tiny value, and the algorithm keeps running until the halting condition is satisfied.

A. Enhanced EBO: In the preceding part, the origin of the EBOwithCMAR framework was introduced. In the following section, modifications to the original equations for EBO perching and patrolling are made to increase the population's variety. Adaptive mechanisms are applied to parameters such as EBO, PS_1, F, CR, and T (T typically represents the current iteration number or time step, indicating the progress stage of the algorithm). At the start of the algorithm, PS_1 and PS_2 populations are initialized by randomly selecting solutions from the entire population X. The improved parts of EBO include the following:

1. Enhanced perching: A new form of perching with modified binomial crossover is introduced as follows:

$$S_{z,j} = \begin{cases} x1_{z,j} + F_{z,j}(x1_{ccz,j} - x1_{z,j} + x1_{r1z,j} - (X1 \cup X2)_{r2z,j}), & \text{if} \\ (\text{rand}_j(0,1) \leq \text{cr}_{z,j} \text{ or } j - j_{\text{rand}}) & \\ x1_{z,j}, & \text{otherwise} \end{cases} \tag{23}$$

2. Enhanced Patrolling: A new form of patrolling with modified binomial crossover is introduced as follows:

$$S_{z,j} = \begin{cases} x1_{z,j} + F_{z,j}(x1_{bestz,j} - x1_{z,j} + x1_{ccz,j} - (X1 \cup X2)_{r2z,j}), & \text{if} \\ (\text{rand}_j(0,1) \leq \text{cr}_{z,j} \text{ or } j - j_{\text{rand}}) & \\ x1_{z,j}, & \text{otherwise} \end{cases} \tag{24}$$

These two equations represent the improved perching mechanism and patrolling mechanism in the EBOwithCMAR framework, utilizing modified binomial crossover methods to enhance the algorithm's exploration and exploitation capabilities. In the perching mechanism, the generation of a new solution $S_{z,j}$ is divided into two cases: if the crossover condition ($\text{rand}_j(0,1) \leq \text{cr}_{z,j}$ or $j = j_{\text{rand}}$) is met, a new solution is generated using the equation, which combines the current solution $x1_{z,j}$, a differential vector $(x1_{ccz,j} - x1_{z,j})$, and other sources $x1_{r1z,j}$ and $(X1 \cup X2)_{r2z,j}$ to maintain diversity; otherwise, the current solution $x1_{z,j}$ is retained. This mechanism focuses on expanding the search range and enhancing the algorithm's exploration capability.

The patrolling mechanism is similar to the perching mechanism but incorporates the best solution $x1_{bestz,j}$ found so far to strengthen local exploitation. When the crossover condition is satisfied, the new solution is generated by combining the current solution, the best solution, and other differential solutions; otherwise, the current solution is retained. The key difference lies in the differential strategy: the perching mechanism emphasizes global exploration, while the patrolling mechanism reinforces exploitation by leveraging information around the best solution. Parameters ccz, $r1z$, $r2z$ denote the indices of different individuals involved in the computation, $cr_{z,j}$ is the crossover probability controlling the inclusion of differential information, and $F_{z,j}$ is the scaling factor adjusting the magnitude of the differential vector, enabling a dynamic balance between exploration and exploitation throughout the algorithm's execution.

3. Individual Selection $best_z$: A novel approach to the selection of $best_z$ is suggested. A random selection of the $best_z$ for the z-th person is made from the D best individuals when $PS_1 > 2D$ (dynamic population size higher than 2D). In the absence of this, if $best_z$ is smaller than 2D, a random selection is made from the top 10% of the population.

4. Compute $prob_{pech}$ and $prob_{pat}$: Both $prob_{pat}$ and $prob_{pech}$ start off at 0.5. At the conclusion of each iteration, the improvement rate of the objective function values I_i^{t+1} is determined using Equations (25) and (26) in order to update these probabilities.

$$I_i^{t+1} = \frac{\sum_{z=1}^{PS_1} \max(0, f_z^{t+1} - f_z^t)}{\sum_{z=1}^{PS_1} f_z^t} \tag{25}$$

$$\forall \bar{x}1_z \text{ upadated by} \begin{cases} perching, & \text{if } i = 1 \\ patrolling, & \text{if } i = 2 \end{cases} \tag{26}$$

$$prob_{pearch} = \max(0.1, \min(0.9, \frac{I_1}{I_1 + I_2})) \tag{27}$$

$$prob_{pat} = 1 - prob_{pearch} \tag{28}$$

The update strategy of $\forall \bar{x}1_z$ is determined by $prob_{pech}$ and $prob_{pat}$. Then, $prob_{pech}$ and $prob_{pat}$ are calculated and updated as per Equations (27) and (28).

5. Linear Reduction in PS_1 and PS_2: At the end of each iteration, PS_1 and PS_2 are linearly reduced by eliminating the worst and random individuals, similar to the LSHADE method for reducing populations.

6. Adaptive F, j_{req}, CR, T: Parameter adaptation techniques are employed for adjusting parameter F. For parameters j_{req}, CR, and T, new parameter settings are sampled using SHBA in this process [27].

B. Covariance Matrix Adaptation Retreat (CMAR): To generate new sample solutions, a mean matrix and covariance matrix are used with a randomly distributed basis. New individuals generated by sampling are weighted based on their objective values. Based on these values, an individual is selected from the sampled individuals to form a new individual. The mean matrix and covariance matrix are then computed based on these new individuals to generate offspring.

C. Probabilities $prob_1$ and $prob_2$ are updated: Two parameters are taken into account in the updating of $prob_1$ and $prob_2$: the variety of the population and the caliber of the solutions. Equation (29) computes the normalized quality value at the conclusion of half a cycle as follows:

$$\hat{Q}_i = \frac{f_{cs/2,i}^{best}}{f_{cs/2,1}^{best} + f_{cs/2,2}^{best}} \tag{29}$$

$f_{cs/2,i}^{best}$ represents the best objective function value at the end of half a cycle for the i-th algorithm (EBO or CMAR). The diversity value is calculated as follows in Equation (30):

$$di\hat{v}_i = \frac{div_i}{div_1 + div_2} \tag{30}$$

At the conclusion of half a cycle, div_i indicates the population diversity rate in relation to the optimal solution, where i stands for the first or second algorithm (EBO or CMAR). Then, Equation (31) is used to determine the progress index PI_i:

$$PI_i = (1 - \hat{Q}_i) + di\hat{v}_i, \forall i = 1, 2 \tag{31}$$

Ultimately, Equation (32) is used to compute the probabilities $prob_i$:

$$prob_i = \max(0.1, \min(0.9, \frac{PI_i}{PI_1 + PI_2})), \forall i = 1, 2 \tag{32}$$

Pr1 and Pr2 are set to 1 if the total of PI is zero.

D. Information Sharing: The algorithm identifies the optimal method for each CS cycle by calculating which algorithm (EBO or CMAR) has the highest probability value. Solutions from population X3 are replaced, with the exception of step duration, by random solutions from population X1, if EBO is thought to be the optimal method. The computation of σ is carried out as follows: $\sigma = \sigma_{initial} \times \left(1 - \frac{FES}{FES_{\max}}\right)$. The default settings of the CMAR parameters are reinstated. The worst individual in X1 is swapped out for the best individual in X3, assuming that CMAR is the optimal algorithm for the cycle. This procedure is repeated as sharing information carries over into the next cycle.

2.5. MadDE

MadDE utilizes three mutation strategies as shown in Equations (33)–(35). The first strategy is a variant of the DE/current-to-pbest/1 mutation strategy based on JADE [25]. The second strategy is a variant of the DE/current-to-rand/1 mutation strategy. Both strategies employ an archive. By maintaining an external archive A, some non-selected solutions are stored. This helps reduce the loss of diversity caused by greedy operations, thereby preserving population diversity and increasing the likelihood of avoiding local optima in subsequent evolution.

DE/current-to-pbest/1 + archive

$$V_i = x_i + F_i \cdot (x_{pbest} - x_i) + F_i \cdot (x_{r1} - x_{r3}) \tag{33}$$

DE/current-to-rand/1 + archive

$$V_i = x_i + F_i \cdot (x_{r1} - x_{r3}) \tag{34}$$

DE/weighted-rand-to-qbest/1

$$V_i = F_i \cdot x_{r1} + F_i \cdot F_a \cdot (x_{qbest} - x_{r2}) \tag{35}$$

Here, $X_{qbest,G}$ is a solution randomly selected from the top q

$$q = 2p - p \cdot \frac{FEs}{FEs_{\max}} \tag{36}$$

And F_a is a variable attraction factor, defined as follows:

$$F_a = 0.5 + 0.5 \cdot \frac{FEs}{FEs_{\max}} \tag{37}$$

These tactics seek to progressively strengthen elitism by heightening the appeal of superior solutions. A random selection of X_{r1} and X_{r2} from the population is used to adjust the amount of greed.

During the crossover process, two main types are used: exponential and binomial crossover. Crossover involves supplying components from the donor vector $V_{i,g}$ to the target vector $X_{i,g}$, generating a new trial vector $U_{i,g}$. This crossover mechanism maintains population diversity, providing opportunities for subsequent evolutionary steps. Here, the focus is on the second type, binomial crossover.

Binomial Crossover (BX): a random number is created in [0, 1], and components from the donor vector $V_{i,g}$ are supplied if it is less than or equal to CR; if not, the target vector $X_{i,g}$ stays unaltered. The way it is applied is seen in Equation (38):

$$U_{j,i,G} = \begin{cases} V_{j,i,G}, & \text{if } (\text{rand}_{i,j}[0,1] \leq \text{Cr or } j = j_{rand}) \\ X_{j,i,G}, & \text{otherwise} \end{cases} \tag{38}$$

q-best Crossover Binomial (qBX): This is a greedy version of the DE/current-to-pbest/1 mutation strategy, based on binomial crossover. In qBX, solutions chosen at random from the top q% of the combined population $P \cup A$ are used to replace the target vector in Equation (38). The calculation of q's value follows the Formula (36). Equation (39) illustrates how these crossover techniques are instantiated probabilistically depending on parameters:

$$\begin{cases} \text{if } (\text{rand}_i[0,1] \leq p_{qBX}), & \text{Perform qBX} \\ \text{otherwise Perform BX} \end{cases} \tag{39}$$

The greediness degree is controlled via parameters in this probabilistic crossover approach. When $p_{qBX} = 0$, it behaves entirely as binomial crossover, whereas when $p_{qBX} = 1$, it performs q-best binomial crossover (qBX). Generally, $p_{qBX} \in [0, 0.5]$.

2.6. IDE-EDA

LSHADE-RSP [32] is a new version of JADE that introduces RSP (Ranking-based Selection Probability) to adjust F_i and CR_i. The mutation strategy based on RSP, DE/current-to-pbest-w/r, is as follows:

$$V_\text{i} = x_i + Fw_i \cdot (x_{pbest} - x_i) + F_i \cdot (x_{pr1} - x_{pr2}) \tag{40}$$

$$Pr_i = Rank_i/(Rank_i + Rank_2 + \ldots + Rank_{NP}) \tag{41}$$

$$Rank_i = k \cdot (NP - i) + 1 \tag{42}$$

In this case, x_{pr1} denotes a vector picked from population P using a rank-based probability and x_{pr2} can be a vector randomly selected from archive A or selected from P using a rank-based probability. With respect to each person i, the selection probability Pr_i is as follows:

Where $Rank_i$ represents the rank of individual i, with higher ranks corresponding to higher probabilities, and NP is the population size. k is a factor that controls the greediness of rank-based selection. Here, Pr_i represents the selection probability for the i-th individual, and $Rank_i$ is the rank of this individual. The ranks $Rank_1, Rank_2, Rank_3, \ldots, Rank_{NP}$ correspond to the ranks of all other individuals in the population, with NP being the total number of individuals. The selection rule calculates Pr_i as the ratio of the individual's rank to the sum of the ranks of all individuals in the population. Typically, lower ranks represent better individuals (e.g., better solutions in an optimization context), so this rule ensures that individuals with lower ranks have a higher probability of being selected, which is commonly used in evolutionary algorithms for reproduction or further operations. Formulas are shown in Equations (40)–(42).

In Equation (43), S denotes S_{CR} or S_F. For each generation, only the k-th cell of MCR and MF is updated. k starts from 1 and increments each generation. Once k exceeds a predefined memory size H, it resets to 1. Initial values for all cells in MF and MCR are set to 0.3. Additionally, throughout evolution, historical memory entries MCR_H and MF_H are fixed at 0.9. This formula computes the weighted mean of the set S. In this calculation, each element S_n is weighted according to ω_n. The numerator represents the sum of the squared values of S_n weighted by ω_n, while the denominator is the weighted sum of S_n itself. The resulting value is a weighted mean, which is typically used to aggregate data based on the importance or priority assigned to each element.

Formula (44) defines the weight ω_n for each element n. The weight is computed as the absolute difference between the fitness values $f(\boldsymbol{u}_n)$ and $f(\boldsymbol{x}_n)$. These fitness values correspond to two different vectors $\boldsymbol{u}_n$ and $\boldsymbol{x}_n$, which may represent different solutions or individuals in an optimization problem. The weight is normalized by dividing the absolute difference by the total sum of such differences across all elements, ensuring that the total sum of weights equals 1.

Formula (45) defines the update rule for $M_{F,k}$. If the set $\mathbf{S}_F$ is empty ($\mathbf{S}_F = \varnothing$), then $M_{F,k}$ remains unchanged. Otherwise, $M_{F,k}$ is updated by taking the average of its current value and the weighted mean of $\mathbf{S}_F$, which is computed using the function $\text{mean}_{WL}(\mathbf{S}_F)$. This rule ensures that $M_{F,k}$ is adjusted iteratively based on the current state of the set $\mathbf{S}_F$.

Formula (46) defines the update rule for $M_{CR,k}$. If $M_{CR,k}$ is in an invalid state ($M_{CR,k} = \perp$) or the maximum value in the set $\mathbf{S}_{CR}$ is zero ($\max(\mathbf{S}_{CR}) = \mathbf{0}$), then $M_{CR,k}$ is set to zero. Otherwise, $M_{CR,k}$ is updated by averaging its current value with the weighted mean of the set $\mathbf{S}_{CR}$, computed as $\text{mean}_{WL}(\mathbf{S}_{CR})$. This rule ensures that $M_{CR,k}$ is either reset to zero or updated based on the state of $\mathbf{S}_{CR}$.

Here, a vector taken from population P using a rank-based probability is indicated by x_{pr1}, and a vector randomly selected from archive A or from P using a rank-based probability can be indicated by x_{pr2}. For every person i, the selection probability Pr_i is as follows:

$$\text{mean}_{WL}(S) = \frac{\sum_{n=1}^{|S|} \omega_n \cdot S_n^2}{\sum_{n=1}^{|S|} \omega_n \cdot S_n} \tag{43}$$

$$\omega_n = \frac{|f(\boldsymbol{u}_n) - f(\boldsymbol{x}_n)|}{\sum_{n=1}^{|S|} |f(\boldsymbol{u}_n) - f(\boldsymbol{x}_n)|} \tag{44}$$

$$M_{F,k} = \begin{cases} M_{F,k}, & \mathbf{S}_F = \varnothing \\ \frac{(\text{mean}_{WL}(\mathbf{S}_F) + M_{F,k})}{2}, & \text{otherwise} \end{cases} \tag{45}$$

$$M_{CR,k} = \begin{cases} \mathbf{0}, & M_{CR,k} = \perp \text{ or } \max(\mathbf{S}_{CR}) = \mathbf{0} \\ \frac{(\text{mean}_{WL}(\mathbf{S}_{CR}) + M_{CR,k})}{2}, & \text{otherwise} \end{cases} \tag{46}$$

$$F_i = \text{randc}(M_{F,R_i}, 0.1) \tag{47}$$

$$Cr_i = \text{randn}(M_{CR,R_i}, 0.1) \tag{48}$$

The F_i and CR_i generated in Equations (47) and (48) are adjusted according to Equations (49) and (50) during the evolutionary process:

$$F_i = \begin{cases} \text{Apply Equation}(45), & F_i \leq 0 \\ \min(F_i, 0.7), & FEs < 0.6 \cdot FEs_{\max} \text{ and } F_i > 0 \\ \min(F_i, 1), & 0.6 \cdot FEs_{\max} \leq FEs \text{ and } F_i > 0 \end{cases} \tag{49}$$

$$Cr_i = \begin{cases} 0, & Cr_i < 0 \\ \max(Cr_i, 0.7), & FEs < 0.25 \cdot FEs_{\max} \text{ and } Cr_i > 0 \\ \max(Cr_i, 0.6), & 0.25 \cdot FEs_{\max} \leq FEs < 0.5 \cdot FEs_{\max} \text{ and } Cr_i > 0 \\ \min(Cr_i, 1), & 0.5 \cdot FEs_{\max} \leq FEs \text{ and } Cr_i > 0 \end{cases} \tag{50}$$

Lastly, the LSHADE-RSP parameter values are as follows: All values in MF are initialized to 0.3, k in Equation (42) is set to 3, and p_{max} in Equation (51) is set to 0.17.

$$P = P_{\min} + (P_{\max} - P_{\min}) \cdot \frac{FES}{FES_{\max}} \tag{51}$$

Two phases are established in the IDE-EDA framework, which is based on the traits of LSHADE-RSP and EDA [33,36–38] (EDA algorithm steps are introduced in Section 3): First, executing the standard LSHADE-RSP algorithm; second, selecting dominant populations from the population to establish an EDA probability distribution model and using the EDA algorithm to generate new solution individuals. The new solution individuals are ranked with the originally generated solution individuals, and the corresponding population size (NP) of solutions from LSHADE-RSP is selected to form a new population. This approach leverages the strong exploitative nature represented by the rank-based mutation selection strategy of LSHADE-RSP, while also fully utilizing the strong exploratory nature of EDA during evolution. By combining these two approaches, deficiencies in orthogonal improvement directions and evolution directions of EDA are mitigated, and EDA helps alleviate the potential loss of diversity in LSHADE-RSP leading to local optima. The interaction between the populations of the two algorithms is the key source of uniqueness in this cooperative evolution system. A greedy approach is employed in maintaining the outside archive A during the evolution process, shrinking it in size as the population NP declines.

2.7. APSM-jSO

The performance of the jSO method is greatly influenced by the crossover factor CR and scaling factor F [30]. The selection of MCR and MF in jSO follows the approach from SHADE, which limits the frequent application of excellent entries during evolution. To address this, an improvement mechanism called APSM (Adaptive Probability Selection Mechanism) is proposed [31].

In APSM, probabilities of entries in MCR and MF are updated based on historical utility information from the previous generation. The specific formulas are given by Equations (52) and (53):

$$PR_h = \frac{SR_h}{\sum_{h=1}^{H} SR_h}, \quad h = 1, 2, \ldots, H \tag{52}$$

$$SR_h = \frac{SN_h}{N_h}, \quad h = 1, 2, \ldots, H \tag{53}$$

The value of N_h represents the number of times the h-th entry in MCR (or MF) is called upon to construct F_i and CR_i. The quantity of successful uses of the h-th item is shown by SN_h, indicating that u_i created using them is superior to x_i. Additionally, the maximum SR is the success rate SR_h of recently modified items in MCR and MF. In the event that every trial participant underperforms their corresponding target participant, all probability of entries in MCR and MF are reset to $1/H$. It is worth noting that only entries from 1 to $H - 1$ in MF and MCR are updated in the evolutionary procedure outlined in this study.

Moreover, the method uses the First-In-First-Out (FIFO) approach. In the original jSO framework, the size of the outside archive $|A|$ is set to $NP \times Arate$. As the population size NP decreases during evolution, $Arate$ remains constant, causing the outside archive size $|A|$ to also decrease. Therefore, in the original jSO framework, some individuals need to be discarded from the archive to store new individuals. Because the maintenance of the external archive in the original jSO framework uses a random selection mechanism, newly entered individuals are combined with existing individuals, and then a subset of size $|A|$ is randomly selected as the archive. Because of this, certain individuals in the outside archive might not be deleted for a number of generations, which would lead to the overuse of some materials. Hence, APSM-jSO introduces a FIFO strategy where newly entered individuals replace the oldest individuals.

Finally, applying the RSP (Ranking-based Selection Probability) method from LSHADE-RSP improves the APSM-jSO algorithm's exploitation potential. However, unlike the original algorithm, this mechanism is used only for the selection of x_{pr1}. Here, x_{r2} is an individual chosen at random from the set of A and P; it is not the same as X_i, X_{pbest}, or X_{pr1}. On one hand, the advantage of randomly selecting from A and P allows the external archive to fully leverage its strengths, particularly in the later stages of evolution, thereby effectively improving the variety of the population. On the other hand, under the selection pressure of the RSP ranking, x_{pr1} is more likely superior to x_{r2}, promoting evolution in a more favorable direction. The strategy is represented as follows:

$$V_i = x_i + Fw_i \cdot (x_{pbest} - x_i) + F_i \cdot (x_{pr1} - x_{r2}) \tag{54}$$

where F_{wi} ranges as described in Equation (55):

$$F_{wi} = \begin{cases} 0.7 \cdot F_i, & \text{FEs} < 0.2 \cdot FEs_{\max} \\ 0.8 \cdot F_i, & 0.2 \cdot FEs_{\max} \le \text{FEs} < 0.4 \cdot FEs_{\max} \\ 1.2 \cdot F_i, & \text{FEs} \ge 0.4 \cdot FEs_{\max} \end{cases} \tag{55}$$

3. The Proposed NLAPSMjSO-EDA Algorithm

In recent years, significant research efforts have focused on adaptive control parameter settings and mutation strategy modifications for DE algorithms. The jSO method was proposed by Brest in 2017 and performed well in the IEEE CEC 2017 competition [2]. Still, jSO's adaptive mechanism has not reached its full optimization potential. The introduction of APSM-jSO represents a significant improvement to the jSO algorithm [31], incorporating straightforward yet effective modifications in scale factor selection, crossover strategy, and external archive updating mechanisms, thereby substantially enhancing the algorithm's performance.

However, the adoption of a linear population reduction strategy significantly limits the overall evolutionary iterations of the population. As a variant of jSO, APSM-jSO inherits the Linear Population Size Reduction (LPSR) scheme from L-SHADE. This approach adjusts the population size linearly across generations. However, such a reduction strategy can lead to an inefficient utilization of computational resources. Maintaining a relatively larger population size in later generations, where exploitation is prioritized, increases the likelihood of resource waste, as fewer meaningful improvements are typically found in this phase. To address this, incorporating a nonlinear population size reduction mechanism might balance exploration and exploitation more effectively, ensuring resources are allocated to areas with higher optimization potential. This results in a higher number of inferior individuals remaining, and since the total number of iterations is fixed, these inferior individuals will also consume iteration counts for their evolution. Therefore, we need to allocate more iterations to the elite population to foster the emergence of superior individuals. The nonlinear population reduction strategy can achieve this [39]. Additionally, we believe that under limited computational resources, moderately integrating EDA algorithms could enhance the performance of the algorithm in high-dimensional spaces [40,41].

Therefore, we have introduced an EDA (Estimation of Distribution Algorithm) to sample and generate individuals from the elite population, aiming to produce higher-quality individuals to lead the iterative evolution of the population. Considering the diversity of the algorithm, we increased the number of individuals in the initial population in subsequent experiments to ensure the diversity of the early population, while the total number of iterations remained unchanged.

In order to address these problems, this work uses the most recent iteration of jSO, APSM-jSO, and suggests two key enhancements:

1. Non-linear population size reduction (NL): NLPSR was first introduced by AGSK in [42] and subsequently employed in [34]. This method suggests using smaller population sizes per generation under fixed maximum FEs, allowing the algorithm more generations to potentially converge to the optimal value. Therefore, employing a non-linear population reduction method enhances the algorithm's capability to explore and exploit, thereby improving both local and global search abilities. The formulas for linear and non-linear population size reduction are shown in Equations (56) and (57), respectively:

$$NP_{L,G+1} = \text{round}\left(\frac{FES}{FES_{\max}} \cdot (NP_{\min} - NP_{\text{init}}) + NP_{\text{init}}\right) \tag{56}$$

$$NP_{NL,G+1} = \text{round}\left(\left(\frac{FES}{FES_{\max}}\right)^{1-\frac{FES}{FES_{\max}}} \cdot (NP_{\min} - NP_{\text{init}}) + NP_{\text{init}}\right) \tag{57}$$

A comparison between linear and non-linear decreases in population size is illustrated in Figure 1, showing that non-linear reduction allows more generations for evolution under a fixed total number of iterations.

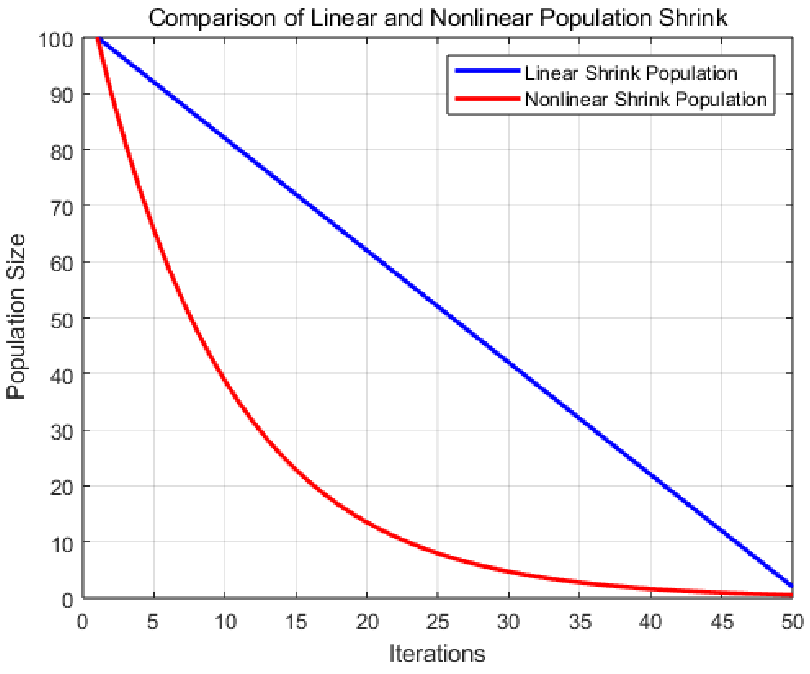

Figure 1. Comparing approaches for reducing population numbers that are linear and non-linear.

Figure 2 illustrates the changes in the ellipsoidal probability density function (PDF) in EDA.

2. Generating advantageous individuals through Gaussian model sampling (EDA). The following are the standard EDA steps:

Step 1: Create M people at random to serve as the starting population.

Step 2: Determine each person's level of fitness for the g-th generation population and review the terms of termination (g represents the generation, initially g = 0). After each iteration update of the entire population, g is incremented by 1, i.e., g = g + 1. If satisfied, break out of the loop; otherwise, carry on.

Step 3: Create the advantageous subpopulation of the (g + 1)-th generation by choosing the top N (N $\leq$ M) beneficial individuals based on fitness values from the population.

Step 4: Using the favorable subpopulation as a basis, update the probability model.

Step 5: Create a new population (size M) by randomly selecting samples from the probability model, then go back to Step 2.

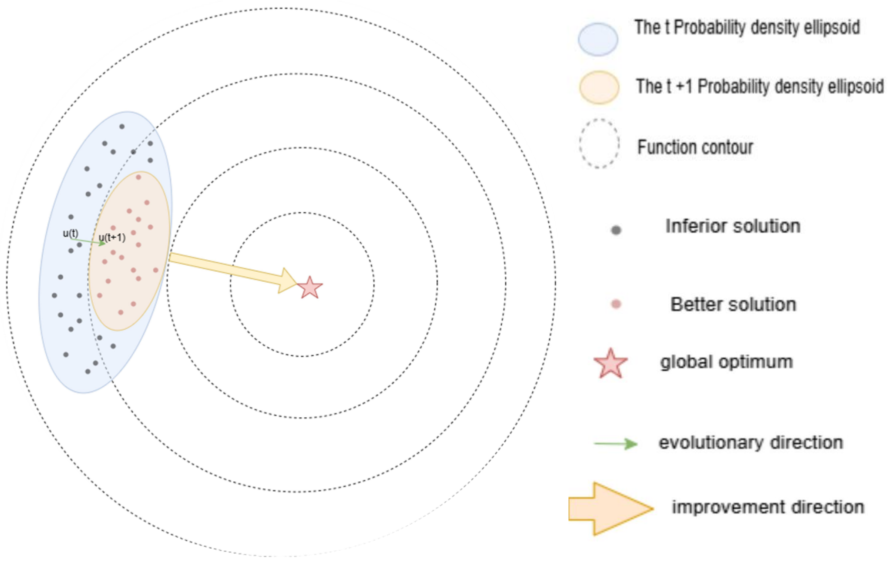

Figure 2. Changes in ellipsoidal probability density function (PDF) in EDA.

The Gaussian model is the most often used model in continuous EDA algorithms [43]. Equation (58) expresses the joint Gaussian density function of D-dimensional random variable X with mean u and covariance matrix C, with the mean and covariance matrix displayed in Equations (59) and (60), respectively:

$$G(x)_{(\mu,C)} = \sqrt{\frac{1}{(2\pi)^D \det(C)}} \exp\left(-\frac{1}{2}(x-\mu)^T C^{-1}(x-\mu)\right) \tag{58}$$

$$\mu = \frac{1}{|P_d|}\sum_{i=1}^{|P_d|} x_i, \quad x_i \in P_d \tag{59}$$

$$C = \frac{1}{|P_d|}\sum_{i=1}^{|P_d|} (x_i-\mu)(x_i-\mu)^T, \quad x_i \in P_d \tag{60}$$

where $|P_d|$ denotes the cardinality of P_d, the set containing advantageous individuals.

As seen in Figure 2, the calculated Gaussian probability model is represented in hyperspace by the PDF's ellipsoid. The mean μ, or the search center of EDA, is represented by the center of the ellipsoid in Figure 2, and the axis directions and lengths of the ellipsoid, which indicate the search direction and range of EDA, are determined by the covariance matrix C.

$$x_i = \mu + g_i, \quad g \sim N(0, C) \tag{61}$$

The advantageous population's estimated Gaussian distribution model is utilized to create a new population using Equation (61). If the vector $x_i$exceeds the search boundary, it is modified using the Formula (62).

$$x_i^j = x_{\min}^j + \text{rand} \cdot (x_{\max}^j - x_{\min}^j), \quad \text{if } x_i^j < x_{\min}^j \text{ or } x_i^j > x_{\max}^j \tag{62}$$

In EDA, the first parameter to consider is NP_{EDA}, i.e., the number of advantageous individuals to sample and compute mean and covariance from the population NP generated by APSM-jSO. Considering that APSM-jSO uses a linear population reduction method (LRSP), even with our enhancement using non-linear population reduction (NLRSP), the magnitude of the population will eventually be down to a minimal amount, leading to insufficient sampling data and reduced diversity. Therefore, we use the following condition in Equation (63) to select NP_{EDA} advantageous individuals for sampling. The EDA algorithm selects half of the individuals with high fitness as beneficial when the population size is larger than or equal to $2 \times D$. When the population size is less than $2 \times D$, all individuals are selected as advantageous individuals for the EDA algorithm.

$$NP_{EDA} = \begin{cases} 0.5 \cdot NP, & \text{if } NP \geqslant 2 \cdot D \\ NP, & \text{if } NP < 2 \cdot D \end{cases} \tag{63}$$

Moreover, NP_{new}, the amount of fresh individuals to be created based on the Gaussian model for selection, is a further variable to take into account. Here, we refer to the parameter τ in the IED-EDA [30] algorithm, with the calculation of NP_{new} as shown in Equation (64). The calculation formula for P is shown in Equation (51).

$$NP_{new} = \tau \cdot P \cdot NP \tag{64}$$

The NLAPSMjSO-EDA algorithm's outline is shown in Figure 3, and Algorithm 1 presents its pseudocode.

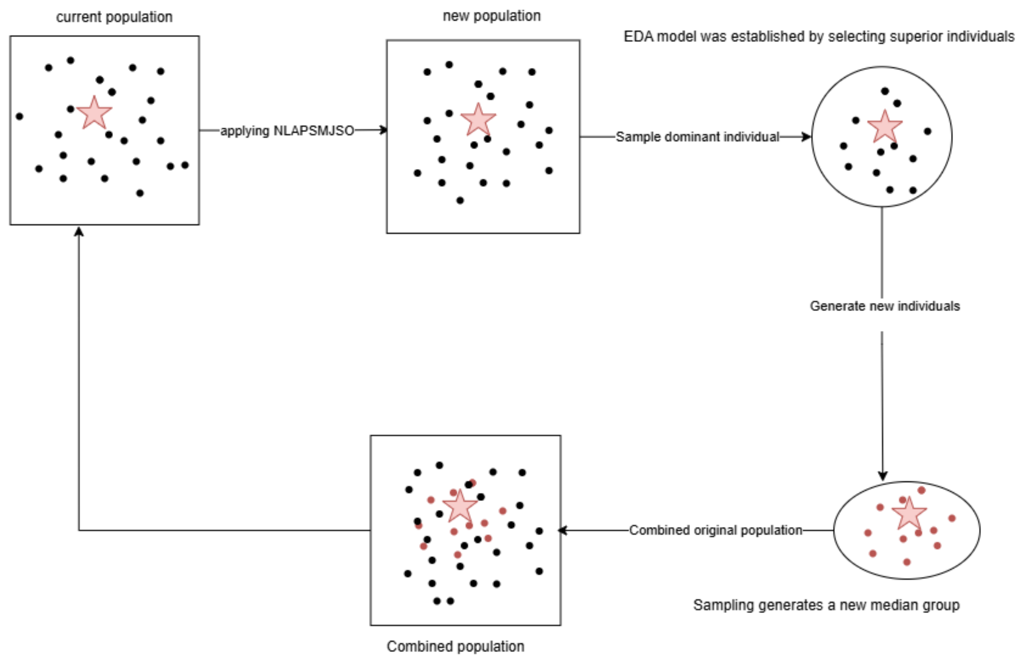
current population
new population
EDA model was established by selecting superior individuals
applying NLAPSMJSO
Sample dominant individual
Generate new individuals
Combined original population
Sampling generates a new median group
Combined population

Figure 3. NLAPSMjSO-EDA program sketch.

Algorithm 1 NLAPSMjSO-EDA algorithm

```
Input: $f(x)$, $FES_{max}$ and solution space
Output: Best solution and objective value to $f(x)$
1: $NP = NP_{init}$, $NP_{init} = 150 \cdot D^{2/3}$, $NP_{min} = 4$, $FEs = 0$, $H = 6$, $P_{max} = 0.17$, $P_{min} = 0.085$, $A = \emptyset$, $nA = 0$, $A_{rate} = 1.3$, $M_{CR} = 0.8$, $M_F = 0.3$, $PR = 1/H$, $M_{F,H} = 0.9$, $M_{CR,H} = 0.9$, $\tau = 0.9$
2: Initialize population $P$ by using (1).
3: Evaluate $P$ to determine their fitness value by using $f(x)$.
4: $FEs = FEs + NP$
5: while $FEs < FES_{max}$ do
6:     Generating $p$ as per (51).
7:     Generating $Pr$ as per (41).
8:     $S_{CR} = \emptyset, S_F = \emptyset$.
9:     for each $i = 1$ in $NP$ do
10:        Generating $F_i$ as per (47) and (49).
11:        Generating $F_{Wi}$ as per (55).
12:        Generating $cr_i$ as per (48) and (50).
13:        Generating mutant vector$v_i$ as per (54).
14:        Checking bound-constraints$v_i$ as per (7).
15:        Generating$U_i$ as per (8).
16:        Evaluating$U_i$and $X_i$ as per (9).
17:        FEs = FEs + 1
18:    end for
19:    for each $i = 1$ in $NP$ do
20:        if $f(u_i) < f(x_i)$ then
21:            Using FIFO strategy and update external archive A
22:                $nA = nA + 1$.
23:                $x_i = u_i$.
24:                $f(x_i) = f(u_i)$.
25:                $S_{CR} = Cr_i \cup S_{CR}$.
26:                $S_F = F_i \cup S_F$.
27:        end if
28:    end for
29:    From population P with population size NP, the dominant population $P_d$ with population size $NP_{EDA}$ is selected to be run by EDA algorithm
30:    Estimate $G(x)(\mu.c)$ by using (59) and (60).
31:    Generating vector $P_E$ as per (61) X with quantity $NP_{new}$ as per (64).
32:    Checking bound-constraints by using (62).
33:    Evaluating$P_E$ fitness value by using $f(P_E)$.
34:    $FEs = FEs + NP_{\text{new}}$
35:    $P = P \cup P_{\text{E}}$
36:    $f(P) = f(P) \cup f(P_{\text{E}})$
37:    Generating PR as per (52).
38:    Updating memory $M_F$ and $M_{CR}$as per (45) and46.
39:    Calculating NP according to (57).
40:    Shrinking P by discarding the worst solutions.
41:    Shrinking A using the FIFO strategy if necessary.
42: end while
```

4. Numerical Trials Utilizing the CEC 2017 Test Suite

In this work, the efficiency of NLAPSMjSO-EDA in 10, 30, 50, and 100 dimensions is examined using the IEEE CEC 2017 test suite [44]. As per the requirements provided by the IEEE CEC 2017 test suite [44], FEs_{max} is set to $10,000 \cdot D$, and the search boundaries for all benchmark functions are $[-100, 100]$. Thus, for $D = 10$, 30, 50, and 100, respectively, FEs_{max} is set to 100,000, 300,000, 500,000, and 1,000,000.

Unless otherwise noted, the findings of every numerical experiment are derived from 51 separate runs. In [44], a thorough explanation of the IEEE CEC 2017 test suite is provided.

On a PC with an AMD Ryzen 7 5700X 8-Core Processor @ 3.60 GHz and 32GB RAM, all tests are carried out in the MATLAB R2020b environment.

The findings document the error fitness value, which is defined as $f(x_{\text{Best}}) - f(x^*)$, where $f(x)$ is the objective function, between x_{Best} and x^*. x^* and x_{Best}, respectively, represent the global optimal solution and the best solution discovered by the issue's approach. For comparative analysis, non-parametric tests like the Friedman and Wilcoxon signed-rank tests are used. The Wilcoxon signed-rank test [45] evaluates the statistical significance between two algorithms for each benchmark function based on 51 separate runs, and the Friedman and Nemenyi tests [45] are used to examine the size of the differences.

The Wilcoxon test [45] is a non-parametric statistical test used to compare the performance of two groups, particularly when the data do not follow a normal distribution. In this study, the Wilcoxon rank-sum test is applied to compare NLAPSMjSO-EDA with each of the six algorithms across multiple test functions. The test determines whether there is a statistically significant difference in performance, with small p-values (e.g., <0.05) indicating significant differences, while larger p-values suggest no significant difference.

The Friedman test, another non-parametric method, is designed to evaluate the performance of more than two algorithms across multiple datasets or functions. It provides an overall comparison of the rankings of all algorithms, including NLAPSMjSO-EDA and its six competitors, on the CEC2017 benchmark functions. If significant differences are detected, post-hoc tests like Holm or Nemenyi can be used to identify specific differences among the algorithms.

The Nemenyi test [45] is a post-hoc statistical method used following a significant Friedman test result to identify specific pairs of algorithms with significant differences in performance rankings. It works by comparing the average ranks of algorithms across multiple datasets or functions, determining significance based on whether the rank differences exceed a critical value derived from the number of algorithms and datasets. In optimization studies, the Nemenyi test is particularly useful for pinpointing which algorithms outperform others after an overall difference has been detected.

Together, these tests offer a comprehensive analysis of the experimental results. The Wilcoxon test highlights pairwise differences between NLAPSMjSO-EDA and other algorithms, while the Friedman test assesses overall performance differences among all competitors. Following a significant Friedman test result, the Nemenyi test provides further insights by identifying which specific pairs of algorithms differ significantly in their performance rankings. This combination of tests ensures a robust and detailed statistical evaluation of the algorithms' effectiveness across multiple benchmark functions.

4.1. Parameter Tuning

Parameter choices have a substantial impact on the efficiency of evolutionary algorithms [46]. Algorithm research usually gives reference values for the parameters that are involved, which the authors have adjusted. Because NLAPSMjSO-EDA introduces the NL nonlinear population reduction method, the algorithm's performance may be impacted by the starting population size NP_{init} (the choice of 2/3 as the exponent for D follows the initial population selection strategy used in APSM-jSO and IDE-EDA). Therefore, we set the parameters as follows, except for NP_{init}: $NP_{\min} = 4$, $k = 3$, $p_{\max} = 0.17$, and $p_{\min} = 0.085$. Four initial values for NP_{init} are tested: $NP_{\text{init}} = 75 \cdot D^{2/3}$, $NP_{\text{init}} = 100 \cdot D^{2/3}$, $NP_{\text{init}} = 125 \cdot D^{2/3}$, and $NP_{\text{init}} = 150 \cdot D^{2/3}$. The performance in 10, 30, 50, and 100 dimensions is compared and analyzed through 51 experiments using the Friedman test.

Parameter tuning was carried out using all functions in the CEC 2017 test suite in order to establish a suitable initial population size. Finding a parameter setting with a comparatively high average ranking was the aim. Take note that this section's experimental results are based on 51 separate runs. Table 3 presents the results of the statistical analysis employing a Friedman test at a level of significance of $\alpha = 0.05$. Figure 4 shows the rankings of NLAPSMjSO-EDA according to the Friedman test for all dimensions.

Table 3. Friedman test rankings for four parameter settings ($\alpha = 0.05$).

NP_{init}	10D	30D	50D	100D	Mean	Mean Rank
$NP_{init} = 75 \cdot D^{2/3}$	2.8966	3.0862	2.9828	3.1034	3.0172	4
$NP_{init} = 100 \cdot D^{2/3}$	**2.2931**	2.2759	**2.2414**	2.4138	2.3060	2
$NP_{init} = 125 \cdot D^{2/3}$	2.3793	**2.2069**	2.4655	2.5172	2.3922	3
$NP_{init} = 150 \cdot D^{2/3}$	2.4310	2.4310	2.3103	**1.9655**	**2.2845**	1

The chi-square for 100D is 11.94, and the p-value is 0.0076; The p-value for 50D is 0.087 and the chi-square is 6.57; the chi-square for 30D is 10.23, and the p-value is 0.0167; the p-value for 10D is 0.1513 and the chi-square is 5.3. Bold values indicate the minimum values in the Friedman ranking.This rule will be followed in subsequent Friedman rankings.

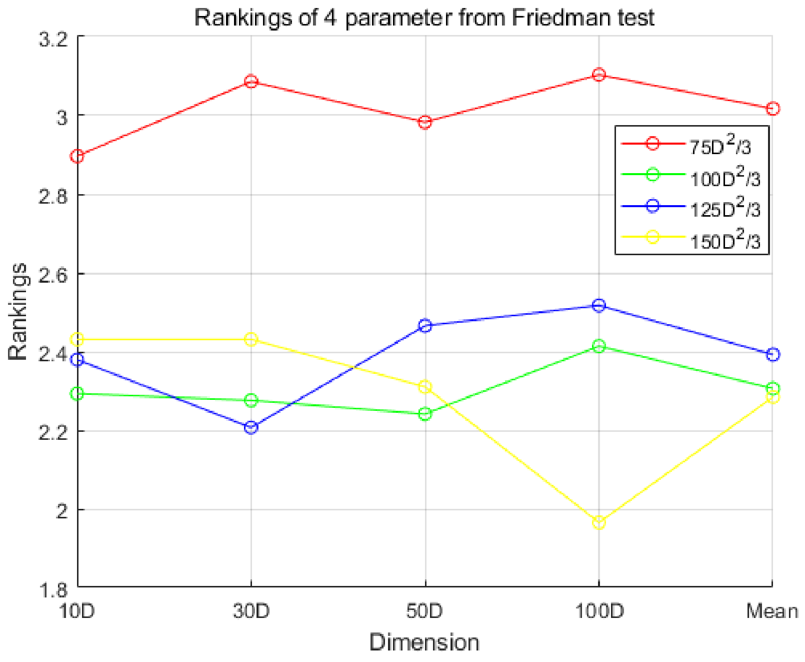

Figure 4. Friedman rankings for four parameter settings.

As shown in the figure, NLAPSMjSO-EDA with $NP_{init} = 100 \cdot D^{2/3}$ achieves the best rankings in 10D and 50D. The setting $NP_{init} = 125 \cdot D^{2/3}$ performs best in 30D, while $NP_{init} = 150 \cdot D^{2/3}$ shows significant improvement in 100D. Regardless, all configurations outperform $NP_{init} = 75 \cdot D^{2/3}$ across various dimensions. Therefore, we select the parameter $NP_{init} = 150D^{2/3}$, which has the highest overall ranking, to maximize the performance of NLAPSMjSO-EDA and use it in subsequent experiments.

4.2. Analysis of Strategy Effectiveness

Two changes are recommended by this research to improve APSM-jSO efficiency. Four distinct versions of the algorithm were developed in order to examine the effects of NL and EDA on the efficiency of APSM-jSO. All four strategies start with an initial population size $NP_{init} = 150 \cdot D^{2/3}$, with other parameters kept identical. NL denotes the use of a nonlinear decreasing strategy, while EDA integrates an EDA algorithm. As shown in Table 4.

We conducted tests on four algorithm variants with different strategies using the CEC2017 benchmark functions. The Friedman ranking was obtained to evaluate the impact of each strategy on the original algorithms and to demonstrate the effectiveness of the proposed strategies in improving performance.

Table 4. Comparison table of four strategies.

Strategy	NLAPSMjSO-EDA	NLAPSM-jSO	APSMjSO-EDA	APSM-jSO
NL	Yes	Yes	NO	NO
EDA	Yes	NO	Yes	NO

To offer a comprehensive examination of the impacts of NL and EDA on APSM-jSO, Table 5 summarizes the Friedman test rankings of NLAPSMjSO-EDA, APSMjSO-EDA, NLAPSM-jSO, and APSM-jSO. “Mean” denotes the average rank across dimensions, whereas “Mean Rank” displays the “Mean” in order of sorting. All p-values across dimensions are smaller than α, as Table 5 illustrates, suggesting that NLAPSMjSO-EDA, APSMjSO-EDA, NLAPSM-jSO, and APSM-jSO perform significantly differently from one another. We show the ranks in Figure 5 to provide a visual representation of them. Based on Table 5 and Figure 5, the following discussions are conducted:

(a) By comparing APSM-jSO with NLAPSM-jSO, we examine the effect of NL’s nonlinear population reduction on APSM-jSO performance. NLAPSM-jSO outperforms APSM-jSO in 30D, 50D, and 100D but performs worse in 10D. Thus, NL can enhance APSM-jSO performance in 30D, 50D, and 100D but may degrade it in 10D.

(b) Comparing APSM-jSO with APSMjSO-EDA allows us to study the impact of integrating EDA on APSM-jSO performance. APSMjSO-EDA performs worse than APSM-jSO across all dimensions (10D, 30D, 50D, 100D), suggesting that EDA does not improve APSM-jSO performance in these dimensions.

(c) Comparing APSM-jSO with NLAPSMjSO-EDA examines the combined impact of NL and EDA on APSM-jSO performance. NLAPSMjSO-EDA outperforms APSM-jSO in 30D, 50D, and 100D but slightly underperforms in 10D. Thus, by employing NL’s nonlinear population reduction and integrating EDA, APSM-jSO performance can be further improved across various dimensions.

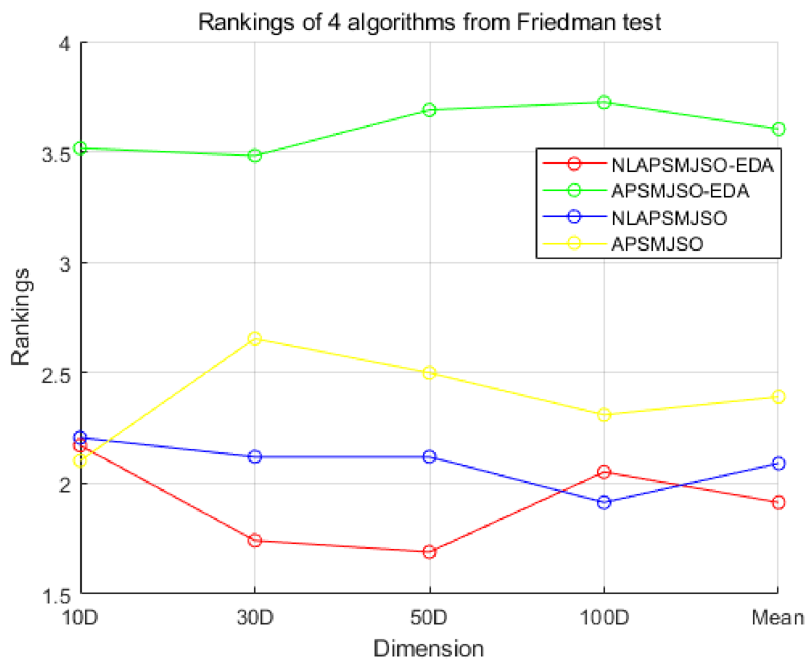

Figure 5. Friedman ranks of four strategies.

Table 5. Friedman ranks of four strategies.

Algorithm	10D	30D	50D	100D	Mean	Mean Rank
NLAPSMjSO-EDA	2.1724	**1.7414**	**1.6897**	2.0517	**1.9138**	1
APSMjSO-EDA	3.5172	3.4828	3.6897	3.7241	3.6034	4
NLAPSM-jSO	2.2069	2.1207	2.1207	**1.9138**	2.0905	2
APSM-jSO	**2.1034**	2.6552	2.500	2.3103	2.3922	3

Chi-square value for 100D is 36.81, with p-value of 5.04×10^{-8}; Chi-square value for 50D is 42.19, with p-value of 3.65×10^{-9}; Chi-square value for 30D is 35.35, with p-value of 1.02×10^{-7}; Chi-square value for 10D is 30.93, with p-value of 8.79×10^{-7}.

The "Mean Rank" is a tool used to assess improvement methods' overall impact on APSM-jSO performance. First place goes to NLAPSMjSO-EDA, then NLAPSM-jSO, APSM-jSO, and APSMjSO-EDA. APSM-jSO and its two versions are outperformed by NLAPSMjSO-EDA, indicating the efficacy of our suggested enhancements and markedly improving APSM-jSO optimization performance. NLAPSMjSO-EDA emerges as a promising variant of APSM-jSO.

To better illustrate the contributions of each strategy to improving the original algorithm (APSM-jSO), the impact is quantified based on the differences in Mean Rank. The Mean Rank of APSM-jSO (2.3922) is used as the baseline. For each strategy, the contribution value is calculated as the difference between the baseline Mean Rank and the strategy's Mean Rank:

$$\text{Contribution Value} = \text{Baseline Mean Rank} - \text{Algorithm Mean Rank}.$$

Positive values indicate improvement, while negative values indicate degradation. To express the contributions as percentages, the contribution values are normalized by dividing each value by the total absolute sum of all contribution values:

$$\text{Contribution Percentage} = \frac{\text{Contribution Value}}{\sum |\text{Contribution Values}|} \times 100\%.$$

This provides a clear and comparable metric for assessing the relative impact of each strategy.

Based on the contributions analysis shown in Table 6, the following conclusions can be drawn:

- NLAPSMjSO-EDA demonstrates the most significant positive contribution, with a Contribution Value of 0.4784 and a Contribution Percentage of 24.04%. This indicates that integrating both nonlinear adjustment and EDA mechanisms effectively improves the baseline algorithm (APSM-jSO).
- APSMjSO-EDA shows a negative Contribution Value of -1.2112, resulting in a Contribution Percentage of -60.86%. This suggests that the exclusive application of EDA degrades the performance compared to the baseline, potentially due to imbalanced parameter exploration and exploitation.
- NLAPSM-jSO achieves a moderate positive Contribution Value of 0.3017 and a Contribution Percentage of 15.15%, indicating that the nonlinear adjustment alone has a beneficial impact on algorithm performance.
- APSM-jSO serves as the baseline algorithm with a Contribution Value of 0, thus having no direct improvement or degradation to compare.

In summary, the combination of nonlinear adjustment and EDA mechanisms in NLAPSMjSO-EDA leads to the most substantial enhancement over the baseline. However, the negative contribution of APSMjSO-EDA highlights the need for careful design and tuning of EDA strategies to avoid performance degradation.

Table 6. Contribution analysis of each algorithm compared to the baseline APSM-jSO.

Algorithm	Mean Rank	Contribution Value	Contribution (%)
NLAPSMjSO-EDA	1.9138	$2.3922 - 1.9138 = 0.4784$	$\frac{0.4784}{1.9908} \times 100\% = 24.04\%$
APSMjSO-EDA	3.6034	$2.3922 - 3.6034 = -1.2112$	$\frac{-1.2112}{1.9908} \times 100\% = -60.86\%$
NLAPSM-jSO	2.0905	$2.3922 - 2.0905 = 0.3017$	$\frac{0.3017}{1.9908} \times 100\% = 15.15\%$
APSM-jSO	2.3922	$2.3922 - 2.3922 = 0$	0%

Additionally, we measure the average population diversity (APD) of NLAPSMjSO-EDA and APSM-jSO. Population diversity is evaluated based on the average population diversity (APD) [47], as shown in Equation (65):

$$DI = \sqrt{\frac{1}{NP}\sum_{i=1}^{NP}\sum_{j=1}^{D}\left(x_{ij} - \overline{x}_j\right)^2}, \quad \overline{x}_j = \frac{1}{NP}\sum_{i=1}^{NP} x_{ij} \tag{65}$$

where $\overline{x}_j$ denotes the mean position of dimension j across the population, and x_{ij} denotes the value of variable j for individual i in the population.

Found in Figures 6–11, the average population diversity (APD) is derived from 51 separate runs. In comparison with algorithms utilizing Estimation of Distribution Algorithms (EDA), the results indicated by the red lines demonstrate a faster decline in diversity compared to the green lines. Similarly, the blue lines exhibit a quicker decrease in diversity than the black lines. This observation confirms that the EDA strategy exhibits a stronger tendency to explore regions near local optima, effectively enhancing local search capabilities. Furthermore, fluctuations in diversity were noted when comparing functions F1, F3, F10, F21, and F23; however, the implementation of the EDA strategy accelerated these fluctuations. Notably, in function F30, the use of the EDA strategy enabled the identification of a better local optimum, resulting in observable disturbances in diversity. These findings validate the efficacy of the EDA strategy in augmenting local search capabilities. In contrast, the NL strategy offers greater opportunities for local exploration.

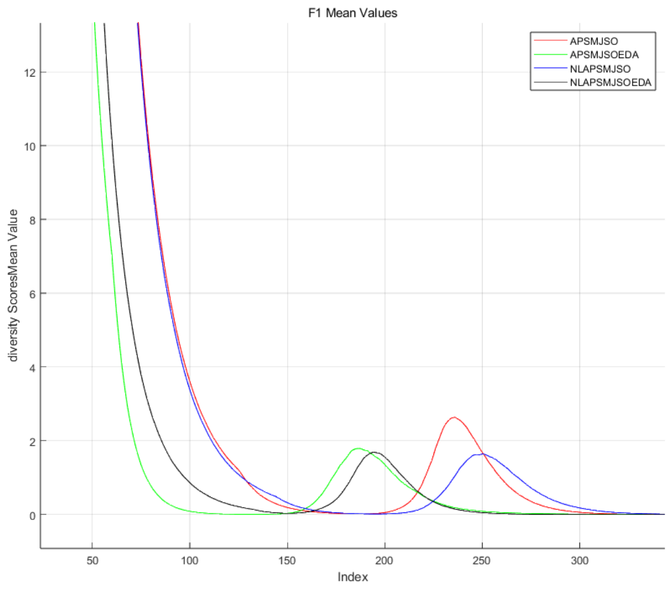

Figure 6. Diversity Comparison of Four Strategies for Function F1 (50D).

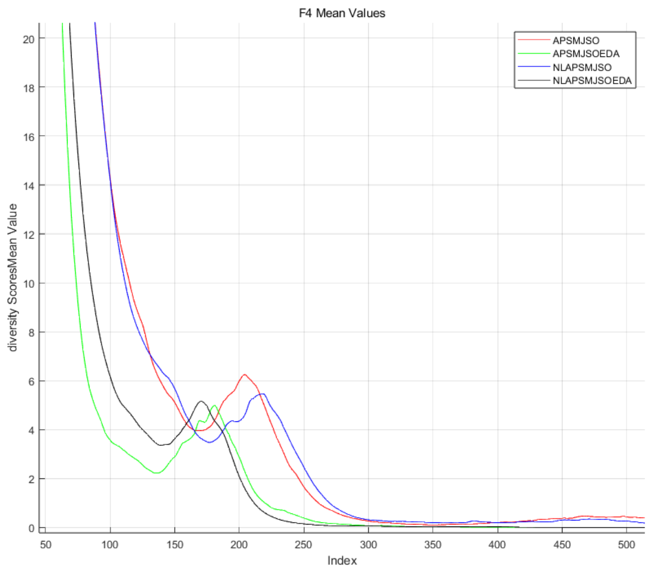

Figure 7. Diversity Comparison of Four Strategies for Function F4 (50D).

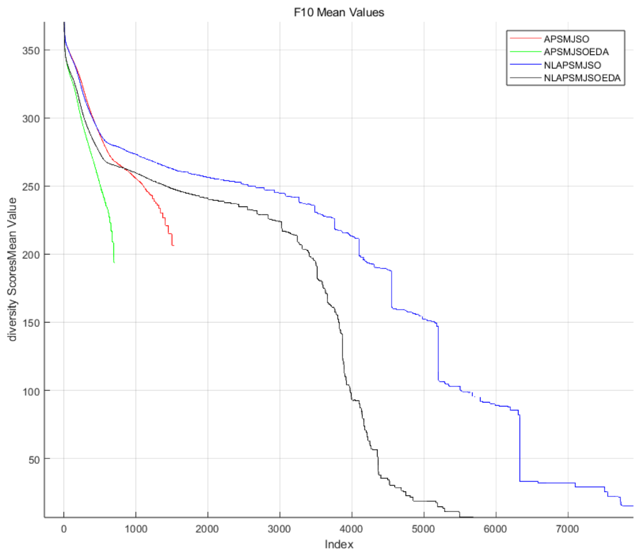

Figure 8. Diversity Comparison of Four Strategies for Function F10 (50D).

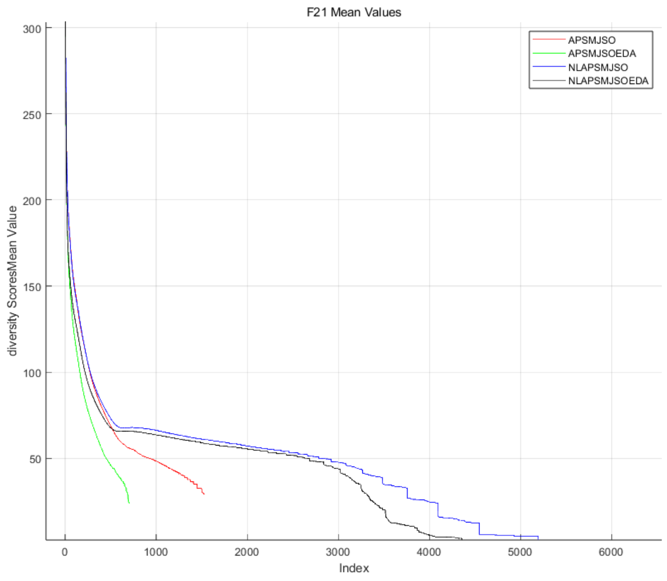

Figure 9. Diversity Comparison of Four Strategies for Function F21 (50D).

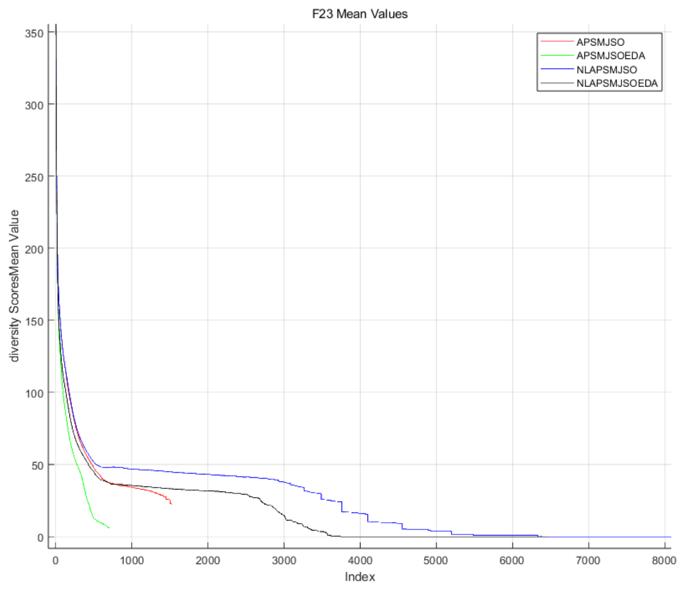

Figure 10. Diversity Comparison of Four Strategies for Function F23 (50D).

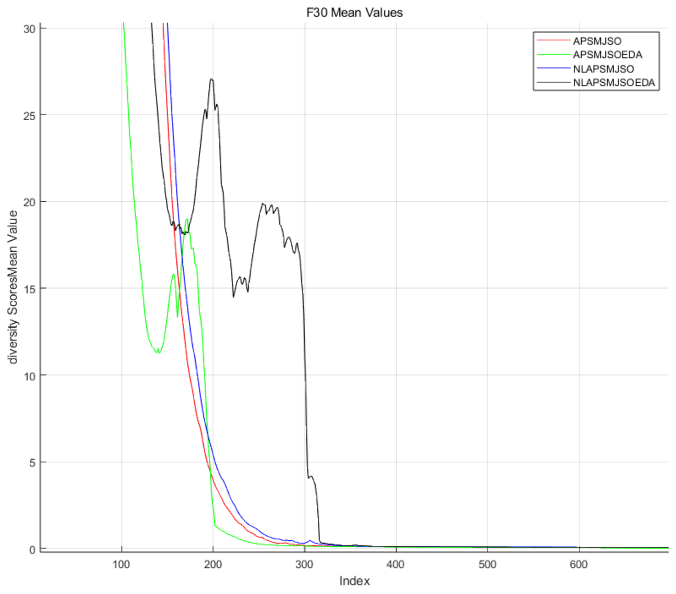

Figure 11. Diversity Comparison of Four Strategies for Function F30 (50D).

4.3. *Compared with Other Advanced DE Algorithms*

The CEC 2017 test suite will be used to assess NLAPSMjSO-EDA's performance in comparison to the most advanced LSHADE or jSO variants, such as LSHADE [27], IDE-EDA [30] (the most recent variant combining LSHADE-RSP and EDA), MadDE [29] (the IEEE CEC 2021 competition winner), APSM-jSO [31] (the most recent variant combining LSHADE-RSP and jSO), EBOwithCMAR [28] (the IEEE CEC 2017 competition champion), and LSHADE-Epsin [26] (third place). To ensure a fair comparison and eliminate any disagreements, the competitors' parameter settings are based on the recommended values from the original literature, as shown in Table 7.

Table 7. Parameter settings of each algorithm.

Algorithm	Parameter Setting
NLAPSMjSO-EDA	$NP_{init} = 150 \cdot D^{2/3}, H = 6, P_{max} = 0.17, P_{min} = 0.085, A_{rate} = 1.3, M_{CR} = 0.8, M_F = 0.3, PR = 1/H, M_{F,H} = 0.9, M_{CR,H} = 0.9, \tau = 0.9$
APSM-jSO [31]	$NP_{init} = 75 \cdot D^{2/3}, k = 3, H = 6, r^{rac} = 1.0, P_{max} = 0.17, P_{min} = 0.085, \tau = 0.9\ , \|A\| = 1.3 \cdot NP$
IDE-EDA [30]	$NP_{init} = 75 \cdot D^{2/3}, k = 3, H = 5, r^{rac} = 1.0, , \tau = 0.9$
LSHADE-Epsin [26]	$NP_{max} = 18 \cdot D, NP_{min} = 4, \|A\| = 1.4 \cdot NP, H = 5, LP = 20, freq = 0.5$, and $G_{ls} = 250$
MadDE [29]	$NP_{init} = 2 \cdot D^2, p_{qBX} = 0.01, p = 0.18, A_{rate} = 2.30, NP_m = 2, H_m = 10, F_0 = 0.20$, and $Cr_0 = 0.20$.
LSHADE [27]	$NP_{max} = 18 \cdot D, NP_{min} = 4, \|A\| = 1.4 \cdot NP; H = 6$, and $p = 0.11$
EBOwithCMAR [28]	$PS1_{max} = 18 \cdot D, PS1_{min} = 4, \sigma = 0.3, H = 6, PS3 = round(4 + 3 \cdot log(D)), CS = 100, 200$, and 300 for the $10D$, $30D$, and $50D$, respectively. $prob1s = 0.1$ and $cfe_{ls} = 0.25 \cdot FEsmax$ as in

4.3.1. Examination of the Wilcoxon Signed-Rank Test Outcomes

The NLAPSMjSO-EDA is evaluated using the CEC 2017 test suite. The results are shown using the Mean (Std.Dev) format. In this case, the labels “Mean” and “Std.Dev” refer to the average and standard deviation of the results.

Tables 8–12 summarize the Wilcoxon signed-rank test findings ($\alpha = 0.05$) for the 51 unique results generated by NLAPSMjSO-EDA, EBOwithCMAR, IDE-EDA, MadDE, APSM-jSO, LSHADE, and LSHADE-Epsin. The characters “+”, “−”, and “=” in Tables 8–11 denote statistically better, poorer, and similar performance of the competitor in relation to NLAPSMjSO-EDA, respectively. Table 13 displays the results of the Wilcoxon rank-sum test, with $p < (a = 0.05)$ denoting a substantial improvement. If not, the improvement is negligible.

Table 8. Comparison Results of NLAPSMjSO-EDA and Six Algorithms on 29 Functions in CEC2017 (Dim = 10).

	APSM-jSO	NLAPSMjSO-EDA	IDE-EDA	LSHADE-Epsin	MadDE	LSHADE	EBOwithCMAR
F1	0.0000×10^0 (0.0000×10^0)	0.0000×10^0 (0.0000×10^0)	0.0000×10^0 (0.0000×10^0)	0.0000×10^0 (0.0000×10^0)	0.0000×10^0 (0.0000×10^0)	0.0000×10^0 (0.0000×10^0)	0.0000×10^0 (0.0000×10^0)
F3	0.0000×10^0 (0.0000×10^0)	0.0000×10^0 (0.0000×10^0)	0.0000×10^0 (0.0000×10^0)	0.0000×10^0 (0.0000×10^0)	0.0000×10^0 (0.0000×10^0)	0.0000×10^0 (0.0000×10^0)	0.0000×10^0 (0.0000×10^0)
F4	0.0000×10^0 (0.0000×10^0)	0.0000×10^0 (0.0000×10^0)	0.0000×10^0 (0.0000×10^0)	0.0000×10^0 (0.0000×10^0)	0.0000×10^0 (0.0000×10^0)	0.0000×10^0 (0.0000×10^0)	0.0000×10^0 (0.0000×10^0)
F5	1.4597×10^0 (7.7508×10^{-1})	9.7587×10^{-1} (6.1302×10^{-1})	9.7593×10^{-1} (7.8322×10^{-1})	1.9147×10^0 (8.6266×10^{-1})	3.8125×10^0 (1.0521×10^0)	2.5183×10^0 (8.5222×10^{-1})	0.0000×10^0 (0.0000×10^0)
F6	0.0000×10^0 (0.0000×10^0)	0.0000×10^0 (0.0000×10^0)	0.0000×10^0 (0.0000×10^0)	0.0000×10^0 (0.0000×10^0)	0.0000×10^0 (0.0000×10^0)	0.0000×10^0 (0.0000×10^0)	1.9509×10^{-2} (1.3932×10^{-1})
F7	1.1551×10^1 (5.2623×10^{-1})	1.1493×10^1 (4.2897×10^{-1})	1.1612×10^1 (5.4942×10^{-1})	1.1949×10^1 (5.7307×10^{-1})	1.4419×10^1 (1.2423×10^0)	1.2196×10^1 (8.1454×10^{-1})	0.0000×10^0 (0.0000×10^0)
F8	1.5095×10^0 (6.4280×10^{-1})	1.2683×10^0 (7.4656×10^{-1})	1.1511×10^0 (7.5442×10^{-1})	1.8566×10^0 (7.6864×10^{-1})	5.0000×10^0 (1.3408×10^0)	2.4401×10^0 (3.9810×10^0)	1.0589×10^1 (2.0689×10^{-1})
F9	0.0000×10^0 (0.0000×10^0)	0.0000×10^0 (0.0000×10^0)	0.0000×10^0 (0.0000×10^0)	0.0000×10^0 (0.0000×10^0)	0.0000×10^0 (0.0000×10^0)	0.0000×10^0 (0.0000×10^0)	0.0000×10^0 (0.0000×10^0)
F10	1.0693×10^1 (2.4166×10^2)	1.6257×10^1 (4.0494×10^2)	1.1320×10^1 (2.3851×10^2)	3.8013×10^1 (5.2839×10^2)	1.0404×10^2 (6.7286×10^2)	2.0402×10^1 (3.3657×10^2)	0.0000×10^0 (0.0000×10^0)
F11	6.3820×10^{-3} (3.1235×10^{-2})	0.0000×10^0 (0.0000×10^0)	0.0000×10^0 (0.0000×10^0)	0.0000×10^0 (0.0000×10^0)	1.4227×10^0 (6.6210×10^{-1})	2.7433×10^{-1} (5.5885×10^{-1})	3.9587×10^1 (5.5759×10^2)

Table 8. *Cont.*

	APSM-jSO	NLAPSMjSO-EDA	IDE-EDA	LSHADE-Epsin	MadDE	LSHADE	EBOwithCMAR
F12	7.3944×10^{0} (2.8401×10^{1})	2.7141×10^{0} (1.6589×10^{1})	2.7355×10^{0} (1.6742×10^{1})	1.1495×10^{2} (5.5348×10^{1})	2.1831×10^{1} (4.5654×10^{1})	2.7060×10^{1} (5.0675×10^{1})	0.0000×10^{0} (0.0000×10^{0})
F13	4.3752×10^{0} (1.6480×10^{0})	3.6244×10^{0} (2.2233×10^{0})	2.8547×10^{0} (2.5608×10^{0})	3.9926×10^{0} (2.6160×10^{0})	2.9518×10^{0} (2.3148×10^{0})	3.8560×10^{0} (2.0549×10^{0})	1.0665×10^{2} (6.1680×10^{1})
F14	1.9514×10^{-2} (1.3932×10^{-1})	0.0000×10^{0} (0.0000×10^{0})	1.9509×10^{-2} (1.3932×10^{-1})	1.1705×10^{-1} (3.8002×10^{-1})	5.8937×10^{-1} (5.3219×10^{-1})	3.4572×10^{-1} (4.8875×10^{-1})	2.8540×10^{0} (2.8306×10^{0})
F15	3.1266×10^{-1} (2.1628×10^{-1})	3.4376×10^{-1} (1.9172×10^{-1})	2.7643×10^{-1} (2.2674×10^{-1})	2.7473×10^{-1} (2.1863×10^{-1})	2.8470×10^{-1} (2.2042×10^{-1})	1.5156×10^{-1} (2.0560×10^{-1})	5.6592×10^{-3} (1.1130×10^{-2})
F16	6.4498×10^{-1} (2.4729×10^{-1})	7.6056×10^{-1} (2.6579×10^{-1})	6.5754×10^{-1} (3.2556×10^{-1})	6.5355×10^{-1} (2.8097×10^{-1})	4.9953×10^{-1} (1.8805×10^{-1})	3.3955×10^{-1} (1.6896×10^{-1})	1.9364×10^{-1} (1.9585×10^{-1})
F17	5.7917×10^{-1} (3.8260×10^{-1})	5.0180×10^{-1} (3.0953×10^{-1})	8.3106×10^{-1} (4.5791×10^{-1})	6.2966×10^{-1} (2.7969×10^{0})	2.7211×10^{-1} (2.5102×10^{-1})	1.3142×10^{-1} (1.5680×10^{-1})	3.9752×10^{-1} (1.7763×10^{-1})
F18	2.2732×10^{-1} (2.0887×10^{-1})	3.7546×10^{-1} (1.6474×10^{-1})	3.4962×10^{-1} (1.8379×10^{-1})	2.6770×10^{0} (6.4939×10^{0})	2.5931×10^{-1} (2.2275×10^{-1})	1.6608×10^{-1} (1.8225×10^{-1})	1.5317×10^{-1} (1.6495×10^{-1})
F19	9.5610×10^{-3} (1.0595×10^{-2})	1.6679×10^{-2} (1.8402×10^{-2})	9.9120×10^{-3} (1.0569×10^{-2})	1.9230×10^{-2} (2.8908×10^{-2})	2.8768×10^{-2} (1.0627×10^{-2})	7.7280×10^{-3} (1.0048×10^{-2})	8.0265×10^{0} (2.7653×10^{0})
F20	3.1217×10^{-1} (1.3961×10^{-1})	3.6726×10^{-1} (1.6168×10^{-1})	4.2235×10^{-1} (1.6309×10^{-1})	2.9993×10^{-1} (2.2477×10^{-1})	0.0000×10^{0} (0.0000×10^{0})	0.0000×10^{0} (0.0000×10^{0})	2.1317×10^{-2} (2.3376×10^{-2})
F21	1.4033×10^{2} (5.0551×10^{1})	1.1623×10^{2} (3.7646×10^{1})	1.4029×10^{2} (5.0664×10^{1})	1.5023×10^{2} (5.1738×10^{1})	1.0181×10^{2} (1.4021×10^{1})	1.5731×10^{2} (5.1566×10^{1})	1.4691×10^{-1} (1.5737×10^{-1})
F22	1.0000×10^{2} (0.0000×10^{0})	1.0000×10^{2} (0.0000×10^{0})	1.0000×10^{2} (0.0000×10^{0})	1.0003×10^{2} (9.7577×10^{-2})	8.8925×10^{1} (2.2536×10^{1})	1.0001×10^{2} (4.0631×10^{-2})	1.2201×10^{2} (4.2028×10^{1})
F23	3.0054×10^{2} (1.1887×10^{0})	3.0023×10^{2} (7.8683×10^{-1})	3.0057×10^{2} (1.1770×10^{0})	3.0187×10^{2} (1.4831×10^{0})	2.8144×10^{2} (8.2934×10^{1})	3.0312×10^{2} (1.6646×10^{0})	1.0000×10^{2} (8.9223×10^{-14})
F24	2.7513×10^{2} (9.8121×10^{1})	2.6131×10^{2} (1.0516×10^{2})	2.8902×10^{2} (8.8373×10^{1})	3.1577×10^{2} (5.4499×10^{1})	9.8039×10^{1} (1.4003×10^{1})	2.9962×10^{2} (7.9080×10^{1})	2.9486×10^{2} (4.2135×10^{1})
F25	4.0513×10^{2} (1.6791×10^{1})	4.0864×10^{2} (1.9445×10^{1})	4.1486×10^{2} (2.2172×10^{1})	4.2212×10^{2} (2.3025×10^{1})	3.9775×10^{2} (4.1010×10^{-2})	4.1410×10^{2} (2.2018×10^{1})	1.5774×10^{2} (9.7721×10^{1})
F26	3.0000×10^{2} (0.0000×10^{0})	3.0000×10^{2} (0.0000×10^{0})	3.0000×10^{2} (0.0000×10^{0})	3.0000×10^{2} (0.0000×10^{0})	1.5490×10^{2} (1.4875×10^{2})	3.0000×10^{2} (0.0000×10^{0})	4.1593×10^{2} (2.2244×10^{1})
F27	3.8948×10^{2} (1.3915×10^{-1})	3.8950×10^{2} (1.0046×10^{-1})	3.8945×10^{2} (1.7810×10^{-1})	3.8913×10^{2} (1.3836×10^{0})	3.8853×10^{2} (7.3392×10^{-1})	3.8947×10^{2} (1.4103×10^{0})	2.8431×10^{2} (3.6729×10^{1})
F28	3.8284×10^{2} (1.3621×10^{2})	3.0611×10^{2} (4.3664×10^{1})	3.0000×10^{2} (0.0000×10^{0})	3.8323×10^{2} (1.2191×10^{2})	2.8235×10^{2} (7.1291×10^{1})	3.2947×10^{2} (9.0380×10^{1})	3.9124×10^{2} (2.2515×10^{0})
F29	2.3401×10^{2} (2.0036×10^{0})	2.3337×10^{2} (2.4295×10^{0})	2.3444×10^{2} (3.7422×10^{0})	2.2863×10^{2} (2.0165×10^{0})	2.4915×10^{2} (6.2155×10^{0})	2.3333×10^{2} (2.4978×10^{0})	3.2618×10^{2} (8.3744×10^{1})
F30	3.9547×10^{2} (6.7405×10^{0})	3.9452×10^{2} (4.4939×10^{-2})	3.9451×10^{2} (2.5976×10^{-2})	1.1928×10^{4} (6.2866×10^{4})	9.5139×10^{2} (1.3874×10^{3})	1.6422×10^{4} (1.1443×10^{4})	3.9543×10^{2} (5.9055×10^{0})
+/=/−	9+/7=/13−	-	10+/8=/11−	5+/8=/16−	14+/5=/10−	8+/6=/15−	14+/4=/11−

Table 9. Comparison Results of NLAPSMjSO-EDA and Six Algorithms on 29 Functions in CEC2017 (Dim = 30).

	APSM-jSO	NLAPSMjSO-EDA	IDE-EDA	LSHADE-Epsin	MadDE	LSHADE	EBOwithCMAR
F1	0.0000×10^{0} (0.0000×10^{0})	0.0000×10^{0} (0.0000×10^{0})	0.0000×10^{0} (0.0000×10^{0})	0.0000×10^{0} (0.0000×10^{0})	2.0085×10^{3} (4.4686×10^{2})	0.0000×10^{0} (0.0000×10^{0})	0.0000×10^{0} (0.0000×10^{0})
F3	0.0000×10^{0} (0.0000×10^{0})	0.0000×10^{0} (0.0000×10^{0})	0.0000×10^{0} (0.0000×10^{0})	0.0000×10^{0} (0.0000×10^{0})	3.0987×10^{4} (1.0225×10^{4})	0.0000×10^{0} (0.0000×10^{0})	0.0000×10^{0} (0.0000×10^{0})
F4	5.8562×10^{1} (0.0000×10^{0})	5.8670×10^{1} (7.7797×10^{-1})	5.8562×10^{1} (0.0000×10^{0})	3.1183×10^{1} (3.8292×10^{0})	8.9962×10^{1} (1.4548×10^{1})	5.8562×10^{1} (2.6662×10^{-14})	5.8888×10^{1} (1.3203×10^{0})

Table 9. *Cont.*

	APSM-jSO	NLAPSMjSO-EDA	IDE-EDA	LSHADE-Epsin	MadDE	LSHADE	EBOwithCMAR
F5	6.7565×10^{0} (1.8352×10^{0})	5.4804×10^{0} (1.3898×10^{0})	7.7081×10^{0} (2.2610×10^{0})	1.1975×10^{1} (2.4555×10^{0})	7.8469×10^{1} (9.6110×10^{0})	6.1353×10^{0} (1.4639×10^{0})	2.6848×10^{0} (1.5856×10^{0})
F6	0.0000×10^{0} (1.0000×10^{-6})	0.0000×10^{0} (0.0000×10^{0})	0.0000×10^{0} (0.0000×10^{0})	4.7399×10^{-7} (7.4314×10^{-7})	1.0989×10^{-1} (3.4848×10^{-2})	2.6838×10^{-9} (1.9166×10^{-8})	0.0000×10^{0} (0.0000×10^{0})
F7	3.7697×10^{1} (1.2529×10^{0})	3.6192×10^{1} (1.0408×10^{0})	3.8789×10^{1} (1.9953×10^{0})	4.2446×10^{1} (2.5291×10^{0})	1.0720×10^{2} (1.2097×10^{1})	3.7334×10^{1} (1.6946×10^{0})	3.3490×10^{1} (8.8282×10^{-1})
F8	7.3926×10^{0} (1.5676×10^{0})	5.8561×10^{0} (1.5547×10^{0})	7.4528×10^{0} (2.2873×10^{0})	1.3141×10^{1} (1.9587×10^{0})	7.2186×10^{1} (8.4910×10^{0})	6.9884×10^{0} (1.4808×10^{0})	2.8330×10^{0} (1.4515×10^{0})
F9	0.0000×10^{0} (0.0000×10^{0})	0.0000×10^{0} (0.0000×10^{0})	0.0000×10^{0} (0.0000×10^{0})	0.0000×10^{0} (0.0000×10^{0})	2.1760×10^{1} (1.4946×10^{1})	0.0000×10^{0} (0.0000×10^{0})	0.0000×10^{0} (0.0000×10^{0})
F10	1.5155×10^{3} (2.3436×10^{2})	1.4653×10^{3} (2.4785×10^{2})	1.8065×10^{3} (2.8564×10^{2})	1.3996×10^{3} (2.4664×10^{2})	2.8562×10^{3} (3.3359×10^{2})	1.4307×10^{3} (2.2902×10^{2})	1.4024×10^{3} (2.4578×10^{2})
F11	1.1380×10^{1} (1.8879×10^{1})	2.5650×10^{0} (8.5401×10^{0})	3.0654×10^{0} (8.3613×10^{0})	1.3413×10^{1} (1.8733×10^{1})	7.3983×10^{1} (1.6706×10^{1})	2.4105×10^{1} (2.7314×10^{1})	6.5152×10^{0} (1.4094×10^{1})
F12	1.9993×10^{2} (1.4808×10^{2})	2.6371×10^{2} (1.5222×10^{2})	1.6374×10^{2} (1.1883×10^{2})	2.6755×10^{2} (1.4290×10^{2})	3.4694×10^{5} (1.3673×10^{5})	1.0163×10^{3} (3.7359×10^{2})	4.0322×10^{2} (2.3769×10^{2})
F13	1.7566×10^{1} (4.7196×10^{0})	1.5568×10^{1} (4.3536×10^{0})	1.5186×10^{1} (6.0672×10^{0})	1.5193×10^{1} (6.6141×10^{0})	1.3548×10^{4} (4.2719×10^{3})	1.4493×10^{1} (5.8338×10^{0})	1.6450×10^{1} (5.0096×10^{0})
F14	2.0727×10^{1} (3.0563×10^{0})	2.0731×10^{1} (7.0364×10^{-1})	2.1231×10^{1} (1.0372×10^{0})	1.9712×10^{1} (5.1714×10^{0})	8.4834×10^{1} (1.7561×10^{1})	2.1051×10^{1} (3.9133×10^{0})	2.2554×10^{1} (2.5901×10^{0})
F15	7.8145×10^{-1} (5.5639×10^{-1})	8.4824×10^{-1} (4.9491×10^{-1})	7.7364×10^{-1} (5.4136×10^{-1})	2.2930×10^{0} (1.5081×10^{0})	3.0360×10^{2} (2.2397×10^{2})	3.2227×10^{0} (1.8159×10^{0})	3.8908×10^{0} (2.1510×10^{0})
F16	3.9764×10^{1} (3.1365×10^{1})	2.6545×10^{1} (3.4830×10^{1})	1.9145×10^{1} (1.9085×10^{1})	1.3066×10^{1} (2.9480×10^{0})	4.6873×10^{2} (1.3235×10^{2})	6.1516×10^{1} (5.9672×10^{1})	5.4572×10^{1} (6.1076×10^{1})
F17	3.4999×10^{1} (6.5204×10^{0})	3.0752×10^{1} (6.6701×10^{0})	3.4884×10^{1} (8.0769×10^{0})	3.2269×10^{1} (5.2834×10^{0})	7.4356×10^{1} (1.4805×10^{1})	3.3008×10^{1} (4.9354×10^{0})	2.9453×10^{1} (6.8529×10^{0})
F18	1.9502×10^{1} (4.8035×10^{0})	2.0635×10^{1} (2.7044×10^{-1})	2.0922×10^{1} (4.4771×10^{-1})	2.0828×10^{1} (5.5874×10^{-1})	6.3282×10^{4} (2.8611×10^{4})	2.2093×10^{1} (1.3765×10^{0})	2.2729×10^{1} (1.5512×10^{0})
F19	3.4884×10^{0} (9.7255×10^{-1})	3.9742×10^{0} (1.0955×10^{0})	3.2951×10^{0} (7.5657×10^{-1})	5.0388×10^{0} (1.4817×10^{0})	2.3781×10^{2} (3.3868×10^{2})	5.1640×10^{0} (1.6073×10^{0})	8.4408×10^{0} (2.2095×10^{0})
F20	3.2112×10^{1} (6.0540×10^{0})	3.1208×10^{1} (4.9836×10^{0})	3.1443×10^{1} (7.5601×10^{0})	3.4842×10^{1} (6.8990×10^{0})	9.9491×10^{1} (4.9076×10^{1})	3.1880×10^{1} (5.3892×10^{0})	3.4757×10^{1} (5.9441×10^{0})
F21	2.0728×10^{2} (1.9872×10^{0})	2.0513×10^{2} (1.5871×10^{0})	2.0689×10^{2} (2.1918×10^{0})	2.1208×10^{2} (2.3964×10^{0})	1.9316×10^{2} (6.8232×10^{1})	2.0732×10^{2} (1.5645×10^{0})	2.0103×10^{2} (1.4510×10^{1})
F22	1.0000×10^{2} (0.0000×10^{0})	1.0000×10^{2} (0.0000×10^{0})	1.0000×10^{2} (0.0000×10^{0})	1.0000×10^{2} (1.7244×10^{-13})	1.0000×10^{2} (1.7148×10^{-4})	1.0000×10^{2} (1.4352×10^{-14})	1.0000×10^{2} (1.4352×10^{-14})
F23	3.4933×10^{2} (3.1823×10^{0})	3.4619×10^{2} (2.8766×10^{0})	3.5047×10^{2} (3.6723×10^{0})	3.5506×10^{2} (4.0714×10^{0})	4.1390×10^{2} (1.0748×10^{1})	3.5006×10^{2} (2.7872×10^{0})	3.5098×10^{2} (3.3913×10^{0})
F24	4.2477×10^{2} (1.7595×10^{0})	4.2258×10^{2} (1.7986×10^{0})	4.2680×10^{2} (2.3096×10^{0})	4.2852×10^{2} (2.4915×10^{0})	4.8637×10^{2} (9.3561×10^{0})	4.2538×10^{2} (1.6907×10^{0})	4.0770×10^{2} (7.0254×10^{1})
F25	3.8670×10^{2} (7.8140×10^{-3})	3.8670×10^{2} (6.5260×10^{-3})	3.8670×10^{2} (4.7190×10^{-3})	3.8664×10^{2} (7.1702×10^{-3})	3.8674×10^{2} (8.3036×10^{-1})	3.8674×10^{2} (2.7818×10^{-2})	3.8649×10^{2} (8.4923×10^{-1})
F26	9.0020×10^{2} (3.4582×10^{1})	8.9164×10^{2} (2.5635×10^{1})	8.9204×10^{2} (3.4491×10^{1})	9.3483×10^{2} (5.2234×10^{1})	2.6863×10^{2} (4.6863×10^{1})	9.3156×10^{2} (4.1785×10^{1})	6.0244×10^{2} (3.0953×10^{2})
F27	4.9830×10^{2} (6.4090×10^{0})	4.9554×10^{2} (7.9651×10^{0})	4.9628×10^{2} (7.5166×10^{0})	5.0308×10^{2} (6.0626×10^{0})	5.1375×10^{2} (3.6290×10^{0})	5.0184×10^{2} (5.4446×10^{0})	5.0323×10^{2} (4.2700×10^{0})

Table 9. *Cont.*

	APSM-jSO	NLAPSMjSO-EDA	IDE-EDA	LSHADE-Epsin	MadDE	LSHADE	EBOwithCMAR
F28	3.2193×10^{2} (4.4885×10^{1})	3.0670×10^{2} (2.7085×10^{1})	3.1522×10^{2} (3.8592×10^{1})	3.1448×10^{2} (3.7607×10^{1})	3.9797×10^{2} (3.5451×10^{0})	3.3513×10^{2} (5.2525×10^{1})	3.1278×10^{2} (3.5396×10^{1})
F29	4.3604×10^{2} (1.2925×10^{1})	4.3337×10^{2} (8.1997×10^{0})	4.4741×10^{2} (2.0588×10^{1})	4.3594×10^{2} (8.4600×10^{0})	5.3511×10^{2} (2.2883×10^{1})	4.3130×10^{2} (6.8252×10^{0})	4.3460×10^{2} (9.8897×10^{0})
F30	1.9700×10^{3} (2.0442×10^{1})	1.9693×10^{3} (9.0305×10^{0})	1.9681×10^{3} (9.4167×10^{0})	1.9712×10^{3} (3.5227×10^{1})	1.2249×10^{4} (3.5520×10^{3})	2.0009×10^{3} (7.4845×10^{1})	1.9925×10^{3} (4.7484×10^{1})
+/=/−	6+/4=/19−	-	7+/4=/18−	6+/3=/20−	2+/0=/27−	5+/3=/21−	9+/5=/15−

Table 10. Comparison Results of NLAPSMjSO-EDA and Six Algorithms on 29 Functions in CEC2017 (Dim = 50).

	APSM-jSO	NLAPSMjSO-EDA	IDE-EDA	LSHADE-Epsin	MadDE	LSHADE	EBOwithCMAR
F1	0.0000×10^{0} (0.0000×10^{0})	0.0000×10^{0} (0.0000×10^{0})	0.0000×10^{0} (0.0000×10^{0})	0.0000×10^{0} (0.0000×10^{0})	1.2010×10^{4} (5.4914×10^{3})	0.0000×10^{0} (0.0000×10^{0})	0.0000×10^{0} (0.0000×10^{0})
F3	0.0000×10^{0} (0.0000×10^{0})	0.0000×10^{0} (0.0000×10^{0})	0.0000×10^{0} (0.0000×10^{0})	0.0000×10^{0} (0.0000×10^{0})	1.2439×10^{5} (1.0812×10^{4})	0.0000×10^{0} (0.0000×10^{0})	2.4009×10^{-10} (1.7146×10^{-9})
F4	5.5393×10^{1} (4.4368×10^{1})	4.6894×10^{1} (3.9567×10^{1})	4.9237×10^{1} (3.9615×10^{1})	5.8050×10^{1} (4.7078×10^{1})	1.1569×10^{2} (2.3266×10^{1})	7.0589×10^{1} (4.8305×10^{1})	4.7009×10^{1} (3.4263×10^{1})
F5	1.4720×10^{1} (3.0272×10^{0})	1.1698×10^{1} (1.8313×10^{0})	1.5473×10^{1} (3.8358×10^{0})	2.6401×10^{1} (6.7155×10^{0})	3.1115×10^{2} (1.7204×10^{1})	1.1932×10^{1} (2.0199×10^{0})	8.1039×10^{0} (2.8686×10^{0})
F6	1.0000×10^{-6} (1.0000×10^{-6})	3.0000×10^{-6} (3.0000×10^{-6})	0.0000×10^{0} (0.0000×10^{0})	7.1807×10^{-7} (5.9741×10^{-7})	2.0683×10^{0} (3.3956×10^{-1})	4.7207×10^{-7} (1.2484×10^{-6})	1.2629×10^{-7} (1.8426×10^{-7})
F7	6.5165×10^{1} (2.2624×10^{0})	6.0569×10^{1} (1.7499×10^{0})	6.9353×10^{1} (3.8844×10^{0})	7.6433×10^{1} (6.4880×10^{0})	3.5319×10^{2} (2.3165×10^{1})	6.3227×10^{1} (1.6055×10^{0})	5.8011×10^{1} (1.2300×10^{0})
F8	1.5509×10^{1} (2.9423×10^{0})	1.2215×10^{1} (2.2012×10^{0})	1.6540×10^{1} (3.6965×10^{0})	2.6879×10^{1} (6.1634×10^{0})	3.0697×10^{2} (1.7207×10^{1})	1.2344×10^{1} (2.1352×10^{0})	9.1497×10^{0} (2.5172×10^{0})
F9	0.0000×10^{0} (0.0000×10^{0})	0.0000×10^{0} (0.0000×10^{0})	0.0000×10^{0} (0.0000×10^{0})	0.0000×10^{0} (0.0000×10^{0})	2.7392×10^{3} (7.8244×10^{2})	1.7555×10^{-3} (1.2536×10^{-2})	0.0000×10^{0} (0.0000×10^{0})
F10	3.2865×10^{3} (3.1859×10^{2})	2.9900×10^{3} (3.4975×10^{2})	3.8357×10^{3} (3.2249×10^{2})	3.1717×10^{3} (3.8363×10^{2})	8.1868×10^{3} (4.9821×10^{2})	3.1687×10^{3} (3.2478×10^{2})	3.1445×10^{3} (4.9974×10^{2})
F11	2.7450×10^{1} (2.7581×10^{0})	2.5050×10^{1} (3.2510×10^{0})	2.3738×10^{1} (3.5157×10^{0})	2.1988×10^{1} (1.6800×10^{0})	3.4092×10^{2} (2.5958×10^{1})	4.8303×10^{1} (1.0170×10^{1})	2.6359×10^{1} (3.2242×10^{0})
F12	1.8097×10^{3} (4.0267×10^{2})	1.8850×10^{3} (4.3245×10^{2})	1.4341×10^{3} (4.3548×10^{2})	1.3060×10^{3} (3.5632×10^{2})	3.2015×10^{6} (4.9104×10^{5})	2.0860×10^{3} (4.9910×10^{2})	2.1117×10^{3} (1.1489×10^{3})
F13	2.9583×10^{1} (2.0744×10^{1})	3.2872×10^{1} (1.8885×10^{1})	3.3759×10^{1} (3.0118×10^{1})	7.1298×10^{1} (3.1669×10^{1})	2.4848×10^{4} (7.9340×10^{3})	6.8967×10^{1} (3.3985×10^{1})	4.2140×10^{1} (2.2268×10^{1})
F14	2.3439×10^{1} (1.6283×10^{0})	2.2885×10^{1} (2.1051×10^{0})	2.3247×10^{1} (1.6316×10^{0})	2.6370×10^{1} (2.0023×10^{0})	1.0228×10^{5} (4.0681×10^{4})	3.0143×10^{1} (2.8262×10^{0})	3.1606×10^{1} (3.7935×10^{0})
F15	2.2697×10^{1} (2.3810×10^{0})	2.2876×10^{1} (2.2918×10^{0})	2.1854×10^{1} (2.0887×10^{0})	2.4569×10^{1} (2.9591×10^{0})	1.6001×10^{4} (2.3481×10^{3})	3.9492×10^{1} (8.3078×10^{0})	2.9178×10^{1} (5.7264×10^{0})
F16	4.1301×10^{2} (1.1377×10^{2})	3.0713×10^{2} (1.3395×10^{2})	3.3154×10^{2} (1.5416×10^{2})	2.7536×10^{2} (1.2406×10^{2})	1.0820×10^{3} (1.7037×10^{2})	4.1221×10^{2} (1.2354×10^{2})	3.6862×10^{2} (1.2392×10^{2})
F17	2.5339×10^{2} (6.4772×10^{1})	2.6388×10^{2} (7.1260×10^{1})	2.8673×10^{2} (8.7716×10^{1})	2.3397×10^{2} (6.3996×10^{1})	7.7755×10^{2} (1.3328×10^{2})	2.3288×10^{2} (6.7878×10^{1})	2.8093×10^{2} (9.5695×10^{1})
F18	2.4661×10^{1} (2.0646×10^{0})	2.3622×10^{1} (1.5667×10^{0})	2.3592×10^{1} (1.9076×10^{0})	2.4927×10^{1} (2.6581×10^{0})	8.0578×10^{5} (2.4935×10^{5})	3.7748×10^{1} (9.6766×10^{0})	3.1987×10^{1} (5.7828×10^{0})
F19	1.3119×10^{1} (2.8199×10^{0})	1.3158×10^{1} (2.3164×10^{0})	1.0881×10^{1} (2.2857×10^{0})	1.7701×10^{1} (2.8843×10^{0})	1.6661×10^{4} (1.5324×10^{3})	2.4099×10^{1} (7.0376×10^{0})	2.4345×10^{1} (3.5351×10^{0})

Table 10. *Cont.*

	APSM-jSO	NLAPSMjSO-EDA	IDE-EDA	LSHADE-Epsin	MadDE	LSHADE	EBOwithCMAR
F20	1.6744×10^{2} (7.2812×10^{1})	1.1559×10^{2} (4.7050×10^{1})	1.4987×10^{2} (7.8564×10^{1})	1.0996×10^{2} (3.5348×10^{1})	5.6374×10^{2} (1.1694×10^{2})	1.5178×10^{2} (4.4205×10^{1})	1.5415×10^{2} (7.4509×10^{1})
F21	2.1457×10^{2} (2.7378×10^{0})	2.1153×10^{2} (2.0945×10^{0})	2.1544×10^{2} (4.1310×10^{0})	2.2690×10^{2} (7.1453×10^{0})	4.7043×10^{2} (1.4674×10^{1})	2.1291×10^{2} (1.9909×10^{0})	2.1290×10^{2} (4.2491×10^{0})
F22	2.1052×10^{3} (1.7718×10^{3})	7.0897×10^{2} (1.3309×10^{3})	1.2822×10^{3} (1.9543×10^{3})	1.6216×10^{3} (1.6989×10^{3})	1.4367×10^{2} (2.8222×10^{1})	2.1285×10^{3} (1.7068×10^{3})	3.0263×10^{2} (8.1718×10^{2})
F23	4.2843×10^{2} (4.7068×10^{0})	4.2158×10^{2} (6.6591×10^{0})	4.3148×10^{2} (5.9473×10^{0})	4.3881×10^{2} (7.3573×10^{0})	7.0828×10^{2} (2.0503×10^{1})	4.2923×10^{2} (4.2074×10^{0})	4.3655×10^{2} (5.8378×10^{0})
F24	5.0487×10^{2} (3.6790×10^{0})	5.0099×10^{2} (3.2923×10^{0})	5.0558×10^{2} (3.7585×10^{0})	5.1379×10^{2} (6.5528×10^{0})	7.6715×10^{2} (1.7173×10^{1})	5.0654×10^{2} (2.2771×10^{0})	5.0755×10^{2} (2.3982×10^{0})
F25	4.8093×10^{2} (2.7535×10^{0})	4.8169×10^{2} (5.9462×10^{0})	4.8215×10^{2} (7.7789×10^{0})	4.8086×10^{2} (2.7551×10^{0})	6.0809×10^{2} (1.1948×10^{0})	4.8382×10^{2} (1.3545×10^{1})	4.8507×10^{2} (1.5280×10^{1})
F26	1.0987×10^{3} (3.7166×10^{1})	1.0511×10^{3} (5.9213×10^{1})	1.0967×10^{3} (5.3759×10^{1})	1.2036×10^{3} (1.0068×10^{2})	3.0645×10^{2} (1.3578×10^{0})	1.1516×10^{3} (5.6520×10^{1})	6.1662×10^{2} (3.8385×10^{2})
F27	5.0853×10^{2} (1.0166×10^{1})	5.0736×10^{2} (7.7208×10^{0})	5.0735×10^{2} (8.6624×10^{0})	5.2841×10^{2} (1.4494×10^{1})	7.1222×10^{2} (2.4731×10^{1})	5.3132×10^{2} (1.7523×10^{1})	5.2366×10^{2} (9.3748×10^{0})
F28	4.6268×10^{2} (1.3263×10^{1})	4.5981×10^{2} (6.8398×10^{0})	4.5885×10^{2} (0.0000×10^{0})	4.5948×10^{2} (1.1709×10^{1})	5.5467×10^{2} (8.6326×10^{0})	4.7576×10^{2} (2.3184×10^{1})	4.6385×10^{2} (1.4677×10^{1})
F29	3.6545×10^{2} (1.0769×10^{1})	3.5537×10^{2} (1.1602×10^{1})	3.8181×10^{2} (2.0988×10^{1})	3.5226×10^{2} (9.7004×10^{0})	1.1263×10^{3} (1.1904×10^{2})	3.5243×10^{2} (1.0687×10^{1})	3.5965×10^{2} (1.9821×10^{1})
F30	6.1178×10^{5} (3.8188×10^{4})	5.8287×10^{5} (1.4549×10^{4})	5.8919×10^{5} (2.3478×10^{4})	6.6199×10^{5} (8.5436×10^{4})	4.0079×10^{6} (6.3440×10^{5})	6.7276×10^{5} (7.3600×10^{4})	6.2112×10^{5} (3.9548×10^{4})
+/=/−	7+/2=/20−	-	8+/2=/19−	10+/2=/17−	2+/0=/27−	2+/2=/25−	6+/1=/22−

Table 11. Comparison Results of NLAPSMjSO-EDA and Six Algorithms on 29 Functions in CEC2017 (Dim = 100).

	APSM-jSO	NLAPSMjSO-EDA	IDE-EDA	LSHADE-Epsin	MadDE	LSHADE	EBOwithCMAR
F1	0.0000×10^{0} (0.0000×10^{0})	0.0000×10^{0} (0.0000×10^{0})	0.0000×10^{0} (0.0000×10^{0})	0.0000×10^{0} (0.0000×10^{0})	2.1054×10^{9} (2.6158×10^{8})	0.0000×10^{0} (0.0000×10^{0})	1.1936×10^{-8} (2.1982×10^{-8})
F3	0.0000×10^{0} (0.0000×10^{0})	0.0000×10^{0} (0.0000×10^{0})	5.0000×10^{-6} (7.0000×10^{-6})	0.0000×10^{0} (0.0000×10^{0})	3.5902×10^{5} (2.1660×10^{4})	1.0381×10^{-6} (1.0381×10^{-6})	2.0227×10^{-5} (8.3569×10^{-6})
F4	1.9494×10^{2} (1.7830×10^{1})	1.9035×10^{2} (3.0388×10^{1})	1.8616×10^{2} (3.0133×10^{1})	1.9907×10^{2} (7.9237×10^{0})	8.4051×10^{2} (4.2385×10^{1})	1.9317×10^{2} (1.9335×10^{1})	1.8333×10^{2} (5.4423×10^{1})
F5	4.1726×10^{1} (4.9713×10^{0})	3.1067×10^{1} (4.0847×10^{0})	5.1883×10^{1} (1.1504×10^{1})	5.4717×10^{1} (6.8995×10^{0})	1.0468×10^{3} (2.8217×10^{1})	3.9267×10^{1} (3.6770×10^{0})	2.8893×10^{1} (4.8578×10^{0})
F6	2.6100×10^{-4} (6.7000×10^{-4})	7.0000×10^{-4} (8.1500×10^{-4})	2.7000×10^{-5} (1.4000×10^{-5})	5.7405×10^{-5} (1.5336×10^{-5})	2.5410×10^{1} (2.8788×10^{0})	6.8190×10^{-3} (4.8034×10^{-3})	1.5502×10^{-5} (6.7106×10^{-6})
F7	1.4426×10^{2} (6.1990×10^{0})	1.2605×10^{2} (3.4714×10^{0})	1.6295×10^{2} (1.0431×10^{1})	1.6215×10^{2} (5.6573×10^{0})	1.1711×10^{3} (3.8832×10^{1})	1.3909×10^{2} (5.1328×10^{0})	1.2155×10^{2} (3.7098×10^{0})
F8	4.2962×10^{1} (5.0001×10^{0})	3.1879×10^{1} (3.8034×10^{0})	5.2367×10^{1} (1.1561×10^{1})	5.5538×10^{1} (1.1238×10^{1})	1.0323×10^{3} (2.7414×10^{1})	3.7470×10^{1} (5.6760×10^{0})	3.0122×10^{1} (5.5578×10^{0})
F9	1.2419×10^{-2} (6.5510×10^{-2})	3.1598×10^{-2} (5.3188×10^{-2})	1.0287×10^{-1} (1.7963×10^{-1})	0.0000×10^{0} (0.0000×10^{0})	4.4959×10^{4} (2.9529×10^{3})	5.0594×10^{-1} (4.5005×10^{-1})	0.0000×10^{0} (0.0000×10^{0})
F10	9.9772×10^{3} (5.3900×10^{2})	8.6097×10^{3} (5.0696×10^{2})	1.1573×10^{4} (6.1048×10^{2})	1.0305×10^{4} (4.8317×10^{2})	2.4716×10^{4} (5.4244×10^{2})	1.0230×10^{4} (6.4982×10^{2})	9.8544×10^{3} (1.7422×10^{3})
F11	1.1181×10^{2} (3.6692×10^{1})	9.1331×10^{1} (3.0331×10^{1})	8.4828×10^{1} (2.3402×10^{1})	5.9879×10^{1} (3.8462×10^{1})	5.8713×10^{4} (5.5282×10^{3})	4.5169×10^{2} (9.9310×10^{1})	6.5514×10^{1} (2.2478×10^{1})

Table 11. *Cont.*

	APSM-jSO	NLAPSMjSO-EDA	IDE-EDA	LSHADE-Epsin	MadDE	LSHADE	EBOwithCMAR
F12	1.3734×10^{4} (6.1133×10^{3})	1.7672×10^{4} (7.7379×10^{3})	1.5908×10^{4} (8.4677×10^{3})	4.7761×10^{3} (6.9664×10^{2})	2.1713×10^{8} (2.3476×10^{7})	2.1733×10^{4} (8.0538×10^{3})	4.3560×10^{3} (6.9693×10^{2})
F13	1.4710×10^{2} (3.8451×10^{1})	2.1667×10^{2} (5.3916×10^{1})	1.3358×10^{2} (4.0724×10^{1})	1.1058×10^{2} (4.0592×10^{1})	4.6689×10^{4} (9.4467×10^{3})	5.2834×10^{2} (3.6883×10^{2})	2.8679×10^{2} (1.0669×10^{2})
F14	6.3204×10^{1} (1.1469×10^{1})	6.3583×10^{1} (1.1249×10^{1})	4.8601×10^{1} (7.3205×10^{0})	5.0545×10^{1} (7.1128×10^{0})	2.5363×10^{6} (5.4105×10^{5})	2.5861×10^{2} (3.0048×10^{1})	1.3837×10^{2} (2.9338×10^{1})
F15	1.7414×10^{2} (3.5252×10^{1})	1.9228×10^{2} (4.0967×10^{1})	1.3598×10^{2} (3.6218×10^{1})	1.0995×10^{2} (4.1445×10^{1})	2.3982×10^{4} (7.0650×10^{3})	2.5563×10^{2} (4.9458×10^{1})	1.5375×10^{2} (3.1048×10^{1})
F16	1.7344×10^{3} (3.3369×10^{2})	1.3588×10^{3} (2.5016×10^{2})	1.6713×10^{3} (4.0174×10^{2})	1.2035×10^{3} (2.5448×10^{2})	6.4020×10^{3} (2.5452×10^{2})	1.6989×10^{3} (2.5395×10^{2})	1.5455×10^{3} (3.9645×10^{2})
F17	1.1889×10^{3} (2.1246×10^{2})	1.0001×10^{3} (1.7403×10^{2})	1.3132×10^{3} (2.3774×10^{2})	9.4853×10^{2} (1.9887×10^{2})	3.5602×10^{3} (2.0434×10^{2})	1.1651×10^{3} (1.9100×10^{2})	1.1688×10^{3} (2.7815×10^{2})
F18	1.7818×10^{2} (3.6906×10^{1})	1.6367×10^{2} (3.3743×10^{1})	1.3817×10^{2} (3.0013×10^{1})	7.5498×10^{1} (1.8433×10^{1})	2.7533×10^{6} (5.3899×10^{5})	2.2812×10^{2} (4.3136×10^{1})	2.5556×10^{2} (5.5086×10^{1})
F19	1.2112×10^{2} (2.2019×10^{1})	1.3646×10^{2} (2.2338×10^{1})	8.0693×10^{1} (1.5373×10^{1})	5.6083×10^{1} (7.1312×10^{0})	3.2232×10^{4} (1.3641×10^{4})	1.7239×10^{2} (2.4797×10^{1})	1.2393×10^{2} (2.1448×10^{1})
F20	1.4964×10^{3} (2.2073×10^{2})	1.2158×10^{3} (2.0828×10^{2})	1.6036×10^{3} (2.4354×10^{2})	1.0483×10^{3} (1.9606×10^{2})	3.4537×10^{3} (2.2572×10^{2})	1.5024×10^{3} (2.1159×10^{2})	1.5095×10^{3} (2.8897×10^{2})
F21	2.6310×10^{2} (5.3056×10^{0})	2.4855×10^{2} (4.6462×10^{0})	2.5715×10^{2} (1.3399×10^{1})	2.8051×10^{2} (1.2598×10^{1})	1.1198×10^{3} (2.6842×10^{1})	2.5824×10^{2} (5.9881×10^{0})	2.5796×10^{2} (5.4792×10^{0})
F22	1.0634×10^{4} (6.1506×10^{2})	9.6001×10^{3} (5.6253×10^{2})	1.2375×10^{4} (7.5754×10^{2})	1.0774×10^{4} (5.3054×10^{2})	2.5451×10^{4} (2.2021×10^{3})	1.1032×10^{4} (1.5871×10^{3})	1.1467×10^{4} (1.7739×10^{3})
F23	5.6438×10^{2} (1.0977×10^{1})	5.6442×10^{2} (8.0281×10^{0})	5.6737×10^{2} (9.4905×10^{0})	5.9525×10^{2} (1.2459×10^{1})	1.4056×10^{3} (2.4844×10^{1})	5.7047×10^{2} (5.8842×10^{2})	5.7733×10^{2} (1.2023×10^{1})
F24	8.9596×10^{2} (6.3263×10^{0})	8.8337×10^{2} (6.1563×10^{0})	8.9675×10^{2} (8.1714×10^{0})	9.1349×10^{2} (8.7061×10^{0})	1.7680×10^{3} (3.8087×10^{1})	9.0980×10^{2} (7.9294×10^{0})	9.1969×10^{2} (1.4433×10^{1})
F25	7.3119×10^{2} (3.4307×10^{1})	7.3768×10^{2} (3.1478×10^{1})	7.2571×10^{2} (4.5376×10^{1})	6.6960×10^{2} (4.1732×10^{1})	1.5871×10^{3} (4.0363×10^{1})	7.4829×10^{2} (3.2178×10^{1})	7.3274×10^{2} (3.8238×10^{1})
F26	3.1418×10^{3} (8.0749×10^{1})	3.0120×10^{3} (8.9381×10^{1})	3.0960×10^{3} (8.6477×10^{1})	3.1196×10^{3} (2.0652×10^{2})	1.8142×10^{4} (1.4210×10^{3})	3.2897×10^{3} (7.0783×10^{1})	3.0198×10^{3} (8.0864×10^{2})
F27	5.8505×10^{2} (1.7139×10^{1})	5.6703×10^{2} (1.7687×10^{1})	5.7523×10^{2} (1.9574×10^{1})	5.8838×10^{2} (1.5396×10^{1})	1.2576×10^{3} (1.2576×10^{3})	6.3057×10^{2} (2.1779×10^{1})	5.8728×10^{2} (1.7361×10^{1})
F28	5.3251×10^{2} (2.7242×10^{1})	5.3985×10^{2} (2.8746×10^{1})	5.2537×10^{2} (2.6127×10^{1})	5.1001×10^{2} (1.9563×10^{1})	2.0852×10^{3} (8.5242×10^{1})	5.2474×10^{2} (2.4086×10^{1})	5.2095×10^{2} (2.7354×10^{1})
F29	1.2617×10^{3} (1.8323×10^{2})	9.4607×10^{2} (9.7974×10^{1})	1.1206×10^{3} (1.7561×10^{2})	1.1356×10^{3} (1.5407×10^{2})	6.0901×10^{3} (1.9487×10^{2})	1.1689×10^{3} (1.5533×10^{2})	1.2860×10^{3} (2.3983×10^{2})
F30	2.3639×10^{3} (1.6222×10^{2})	2.2193×10^{3} (9.7362×10^{1})	2.1689×10^{3} (9.8707×10^{1})	2.3724×10^{3} (1.7574×10^{2})	3.7075×10^{6} (5.4850×10^{5})	2.3799×10^{3} (1.3355×10^{2})	2.3519×10^{3} (1.2372×10^{2})
+/=/−	7+/2=/20−	−	12+/1=/16−	13+/2=/14−	0+/0=/29−	0+/0=/29−	12+/0=/17−

Table 12. Results of the Wilcoxon signed-rank test for six DE variants and NLAPSMjSO-EDA.

Algorithm	10D	30D	50D	100D	Total
NLAPSMjSO-EDA VS. APSM-jSO	13+/7=/9−	19+/4=/6−	20+/2=/7−	20+/2=/7−	72+/15=/29−
NLAPSMjSO-EDA VS. IDE-EDA	11+/8=/10−	18+/4=/7−	19+/2=/8−	16+/1=/12−	64+/15=/37−
NLAPSMjSO-EDA VS. LSHADE-Epsin	16+/8=/5−	20+/3=/6−	17+/2=/10−	14+/2=/13−	67+/15=/34−

Table 12. *Cont.*

Algorithm	10D	30D	50D	100D	Total
NLAPSMjSO-EDA VS. MadDE	10+/5=/14−	27+/0=/2−	27+/0=/2−	29+/0=/0−	93+/5=/18−
NLAPSMjSO-EDA VS. LSHADE	15+/6=/8−	21+/3=/5−	25+/2=/2−	29+/0=/0−	90+/11=/15−
NLAPSMjSO-EDA VS. EBOwithCMAR	11+/4=/14−	16+/4=/9−	22+/1=/6−	17+/0=/12−	66+/9=/41−

Table 13. *P*-values for Wilcoxon Rank-Sum Tests of NLAPSMjSO-EDA and Six Algorithms.

Algorithm	10D	30D	50D	100D
NLAPSMjSO-EDA VS. APSM-jSO	3.9097×10^{-2}	1.5169×10^{-3}	2.3602×10^{-3}	1.293×10^{-2}
NLAPSMjSO-EDA VS. IDE-EDA	2.744×10^{-1}	1.8828×10^{-2}	2.1796×10^{-3}	1.6516×10^{-1}
NLAPSMjSO-EDA VS. LSHADE-Epsin	1.5258×10^{-3}	1.4401×10^{-2}	1.7241×10^{-2}	4.2615×10^{-1}
NLAPSMjSO-EDA VS. MadDE	3.3672×10^{-1}	2.3467×10^{-5}	2.8208×10^{-5}	1.3159×10^{-6}
NLAPSMjSO-EDA VS. LSHADE	4.5526×10^{-3}	7.3196×10^{-4}	5.6227×10^{-5}	6.315×10^{-6}
NLAPSMjSO-EDA VS. EBOwithCMAR	3.6073×10^{-1}	1.4039×10^{-1}	3.1395×10^{-3}	3.1091×10^{-2}

The discussion of the results from Tables 8–12 is as follows:

(a) For 10D, the advantages (disadvantages) of NLAPSMjSO-EDA compared to APSM-jSO, IDE-EDA, LSHADE-Epsin, MadDE, LSHADE, and EBOwithCMAR in the test functions are 13(9), 11(10), 16(5), 10(14), 15(8), and 11(14), respectively. This means that NLAPSMjSO-EDA outperforms four DE-based competitors (APSM-jSO, IDE-EDA, LSHADE-Epsin, LSHADE) in 10D and is underperformed by MadDE and EBOwithCMAR.

(b) For 30D, the advantages (disadvantages) of NLAPSMjSO-EDA compared to APSM-jSO, IDE-EDA, LSHADE-Epsin, MadDE, LSHADE, and EBOwithCMAR in the test functions are 19(6), 18(7), 20(6), 27(2), 21(5), and 16(9), respectively. This means that NLAPSMjSO-EDA outperforms all six DE-based competitors (APSM-jSO, IDE-EDA, LSHADE-Epsin, MadDE, LSHADE, EBOwithCMAR) in 30D.

(c) For 50D, the advantages (disadvantages) of NLAPSMjSO-EDA compared to APSM-jSO, IDE-EDA, LSHADE-Epsin, MadDE, LSHADE, and EBOwithCMAR in the test functions are 20(7), 19(8), 17(10), 27(2), 25(2), and 22(6), respectively. This means that NLAPSMjSO-EDA outperforms all six DE-based competitors (APSM-jSO, IDE-EDA, LSHADE-Epsin, MadDE, LSHADE, EBOwithCMAR) in 50D.

(d) For 100D, the advantages (disadvantages) of NLAPSMjSO-EDA compared to APSM-jSO, IDE-EDA, LSHADE-Epsin, MadDE, LSHADE, and EBOwithCMAR in the test functions are 20(7), 16(12), 14(13), 29(0), 29(0), and 17(12), respectively. This means that NLAPSMjSO-EDA outperforms four DE-based competitors (APSM-jSO, LSHADE-Epsin, LSHADE, EBOwithCMAR) in 100D and is outperformed by MadDE and EBOwithCMAR.

Table 12's "Total" column shows that, for the test functions, NLAPSMjSO-EDA's benefits (disadvantages) over APSM-jSO, IDE-EDA, LSHADE-Epsin, MadDE, LSHADE, and EBOwithCMAR are 72(29), 64(37), 67(34), 93(18), 90(15), and 65(41), in that order. This indicates that NLAPSMjSO-EDA outperforms all six DE variants.

The discussion of the results from Table 13 is as follows:

(a) For 10D, the *p*-values of the Wilcoxon rank-sum test for NLAPSMjSO-EDA compared to APSM-jSO, IDE-EDA, LSHADE-Epsin, MadDE, LSHADE, and EBOwithCMAR are 0.039097, 0.2744, 0.0015258, 0.33672, 0.0045526, and 0.36073, respectively. This indicates that NLAPSMjSO-EDA is significantly better than three DE-based competitors (APSM-jSO, LSHADE-Epsin, LSHADE) in 10D and shows no significant difference compared to IDE-EDA, MadDE, and EBOwithCMAR.

(b) For 30D, the *p*-values of the Wilcoxon rank-sum test for NLAPSMjSO-EDA compared to APSM-jSO, IDE-EDA, LSHADE-Epsin, MadDE, LSHADE, and EBOwithCMAR are 0.0015169, 0.018828, 0.014401, 2.3467×10^{-5}, 0.00073196, and 0.14039, respectively. This indicates that NLAPSMjSO-EDA is significantly better than five DE-based competitors (APSM-jSO, IDE-EDA, LSHADE-Epsin, MadDE, LSHADE) in 30D and shows no significant difference compared to EBOwithCMAR.

(c) For 50D, the *p*-values of the Wilcoxon rank-sum test for NLAPSMjSO-EDA compared to APSM-jSO, IDE-EDA, LSHADE-Epsin, MadDE, LSHADE, and EBOwithCMAR are 0.0023602, 0.0021796, 0.017241, 2.8208×10^{-5}, 5.6227×10^{-5}, and 0.0031395, respectively. This indicates that NLAPSMjSO-EDA is significantly better than six DE-based competitors (APSM-jSO, IDE-EDA, LSHADE-Epsin, MadDE, LSHADE, and EBOwithCMAR) in 50D.

(d) For 100D, the *p*-values of the Wilcoxon rank-sum test for NLAPSMjSO-EDA compared to APSM-jSO, IDE-EDA, LSHADE-Epsin, MadDE, LSHADE, and EBOwithCMAR are 0.01293, 0.16516, 0.42615, 1.3159×10^{-6}, 6.315×10^{-6}, and 0.031091, respectively. This indicates that NLAPSMjSO-EDA is significantly better than four DE-based competitors (APSM-jSO, LSHADE-Epsin, LSHADE, EBOwithCMAR) in 100D and shows no significant difference compared to IDE-EDA and LSHADE-Epsin.

4.3.2. Analysis of Friedman Test Results

In this part, we evaluate the Friedman test results to investigate the overall performance differences between NLAPSMjSO-EDA and six DE variants. Table 14 summarizes the Friedman test ranks, and Figure 12 shows how they are graphically represented. Table 14 shows that the *p*-values for 30D, 50D, and 100D are all less than α, indicating significant differences between NLAPSMjSO-EDA and the six DE-based versions.

The following provides specific Friedman test results:

(a) For 10D, NLAPSMjSO-EDA ranks first, followed by EBOwithCMAR, IDE-EDA, MadDE, APSM-jSO, LSHADE, and LSHADE-Epsin. This indicates that NLAPSMjSO-EDA outperforms the six DE-based competitors in 10D according to the Friedman test results, though the differences are not significant.

(b) For 30D, NLAPSMjSO-EDA ranks first, followed by IDE-EDA, EBOwithCMAR, APSM-jSO, LSHADE, LSHADE-Epsin, and MadDE. This indicates that NLAPSMjSO-EDA significantly outperforms the six DE-based competitors in 30D according to the Friedman test results.

(c) For 50D, NLAPSMjSO-EDA ranks first, followed by IDE-EDA, APSM-jSO, EBOwithCMAR, LSHADE-Epsin, LSHADE, and MadDE. This indicates that NLAPSMjSO-EDA significantly outperforms the six DE-based competitors in 50D according to the Friedman test results.

(d) For 100D, NLAPSMjSO-EDA ranks first, followed by LSHADE-Epsin, EBOwithCMAR, IDE-EDA, APSM-jSO, LSHADE, and MadDE. This indicates that NLAPSMjSO-EDA significantly outperforms the six DE-based competitors in 100D according to the Friedman test results.

In Table 14, NLAPSMjSO-EDA is ranked first by the "Mean Rank"; IDE-EDA, EBOwithCMAR, APSM-jSO, LSHADE-Epsin, LSHADE, and MadDE are ranked thereafter. We may infer that NLAPSMjSO-EDA performs better than any of its rivals, proving that it greatly improves APSM-jSO performance. A competitive jSO version and a potential APSM-jSO variation is NLAPSMjSO-EDA.

In order to validate the differences between NLAPSMjSO-EDA, EBOwithCMAR, IDE-EDA, MadDE, APSM-jSO, LSHADE, and LSHADE-Epsin based on the Friedman test findings, Nemenyi tests [45] were also employed as post-hoc testing. The extent of differences across dimensions between NLAPSMjSO-EDA and the six DE-based rivals is shown in Figure 13, while the differences in the “Overall dimension-wise comparison ranking” based on Friedman test findings are shown in Figure 14. Significant statistical differences are shown by the red lines.

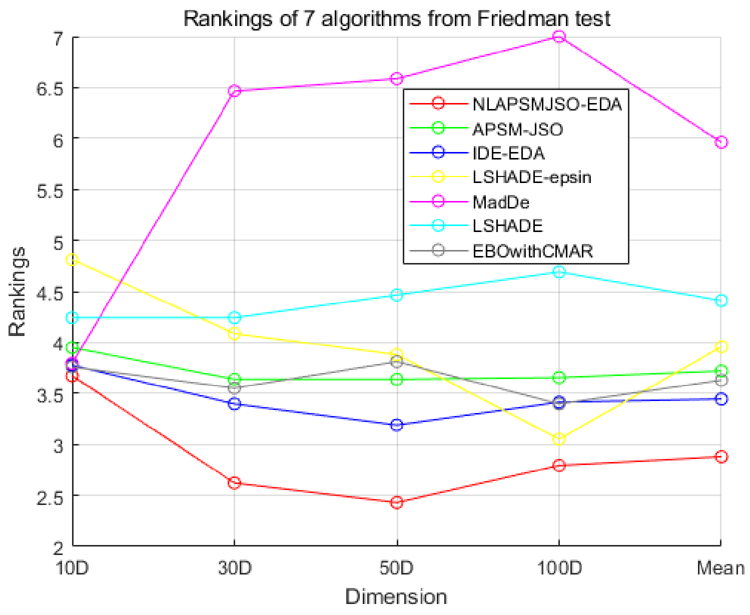

Figure 12. Friedman rankings of the seven algorithms.

Table 14. Algorithm Performance Comparison.

Algorithm	10D	30D	50D	100D	Mean	Mean Rank
NLAPSMjSO-EDA	**3.6724**	**2.6207**	**2.4310**	**2.7931**	**2.8793**	1
APSM-jSO	3.9483	3.6379	3.6379	3.6552	3.7198	4
IDE-EDA	3.7759	3.3966	3.1897	3.4138	3.4440	2
LSHADE-Epsin	4.8103	4.0862	3.8793	3.0517	3.9569	5
MadDE	3.7931	6.4655	6.5862	7	5.9612	7
LSHADE	4.2414	4.2414	4.4655	4.6897	4.4095	6
EBOwithCMAR	3.7586	3.5517	3.8103	3.3966	3.6293	3

At 100D, the chi-square is 79.89 with a p-value of 3.76×10^{-15}. At 50D, the chi-square is 66.49 with a p-value of 2.14×10^{-12}. At 30D, the chi-square is 60.81 with a p-value of 3.08×10^{-11}. At 10D, the chi-square is 7.37 with a p-value of 0.2876.

From Figure 13, NLAPSMjSO-EDA ranks first in 10D, showing no significant differences from its competitors. In 30D, 50D, and 100D, NLAPSMjSO-EDA considerably outperforms MadDE, and in 50D and 100D, it outperforms LSHADE and MadDE, respectively. However, for EBOwithCMAR, IDE-EDA, APSM-jSO, and LSHADE-Epsin, there are no significant differences. Figure 14 indicates that NLAPSMjSO-EDA shows no significant differences in overall dimension-wise comparison rankings compared to IDE-EDA and EBOwithCMAR, but significantly outperforms MadDE, APSM-jSO, LSHADE, and LSHADE-Epsin.

Considering the poor performance of MadDE in 30D, 50D, and 100D, it was excluded, and the Friedman test rankings were recalculated, resulting in Figures 15 and 16. From Figure 15, NLAPSMjSO-EDA ranks first in 10D, showing no significant differences from its competitors. NLAPSMjSO-EDA performs much better than LSHADE and LSHADE-Epsin for 30D and 50D, but it does not vary significantly from EBOwithCMAR, IDE-EDA, or APSM-jSO. In 100D, NLAPSMjSO-EDA significantly outperforms LSHADE. Figure 16 indicates that in the overall dimension-wise comparison rankings, NLAPSMjSO-EDA shows no significant differences compared to IDE-EDA, but significantly outperforms

EBOwithCMAR, APSM-jSO, LSHADE, and LSHADE-Epsin. Overall, NLAPSMjSO-EDA proves to be a highly competitive algorithm.

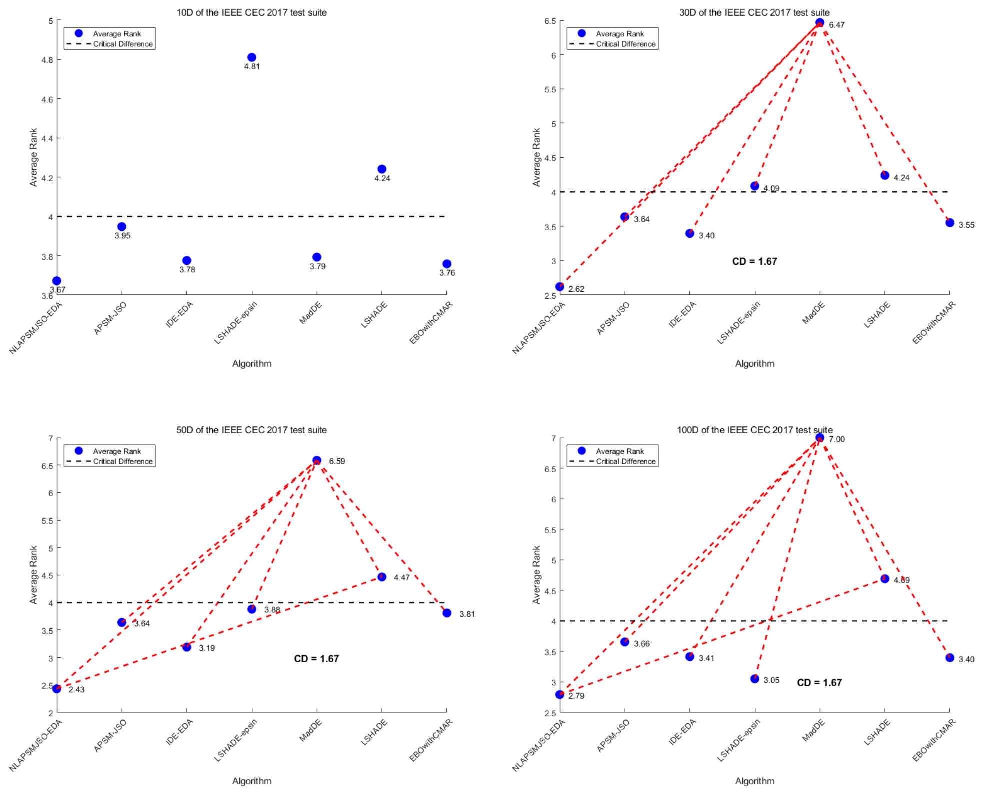

Figure 13. Comparing NLAPSMjSO-EDA and six DE-based rivals dimension-wise with CDV.

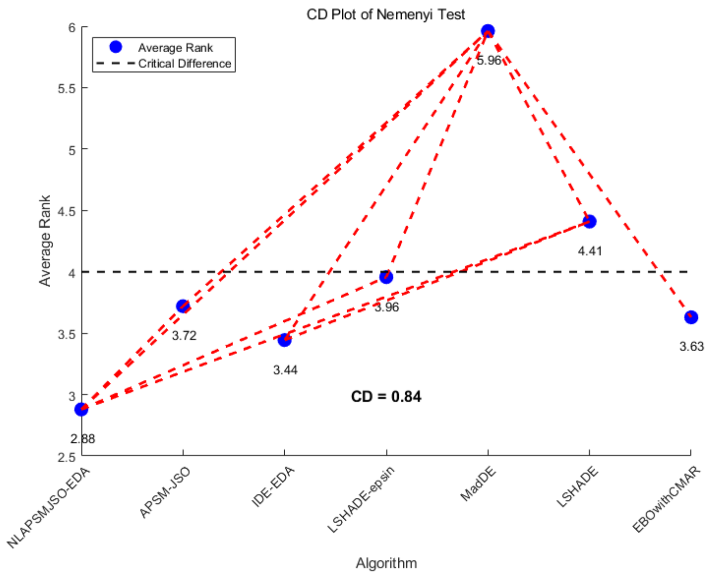

Figure 14. Comparing six DE-based rivals with NLAPSMjSO-EDA overall dimension-wise utilizing CDV.

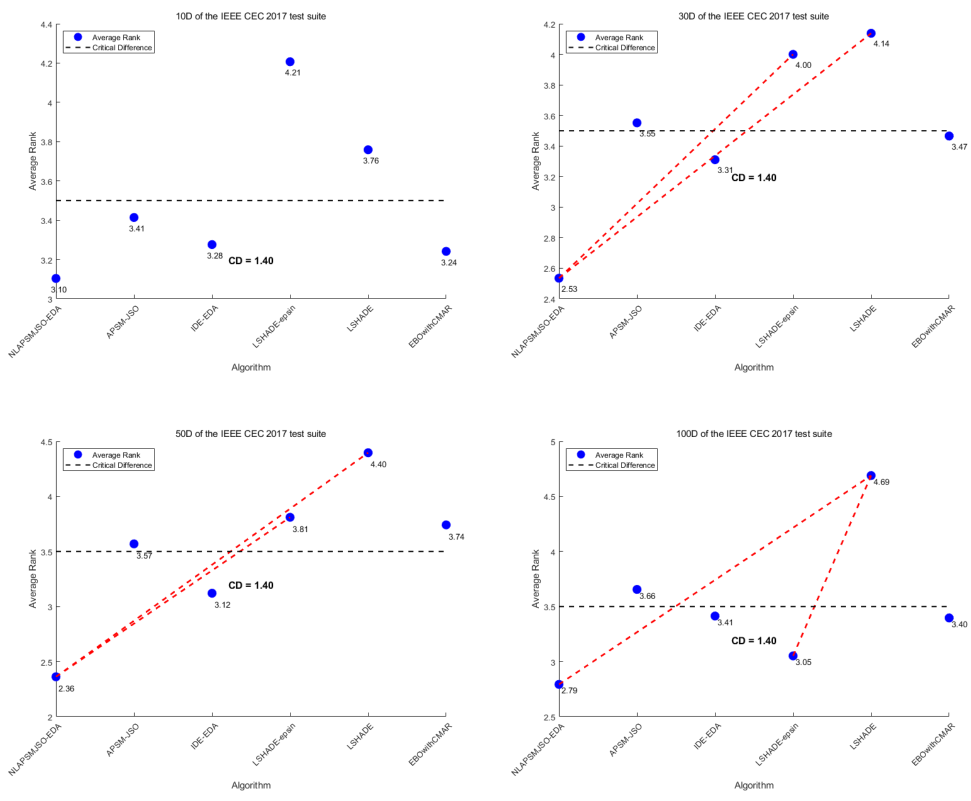

Figure 15. Comparing NLAPSMjSO-EDA and five DE-based rivals dimension-wise with CDV.

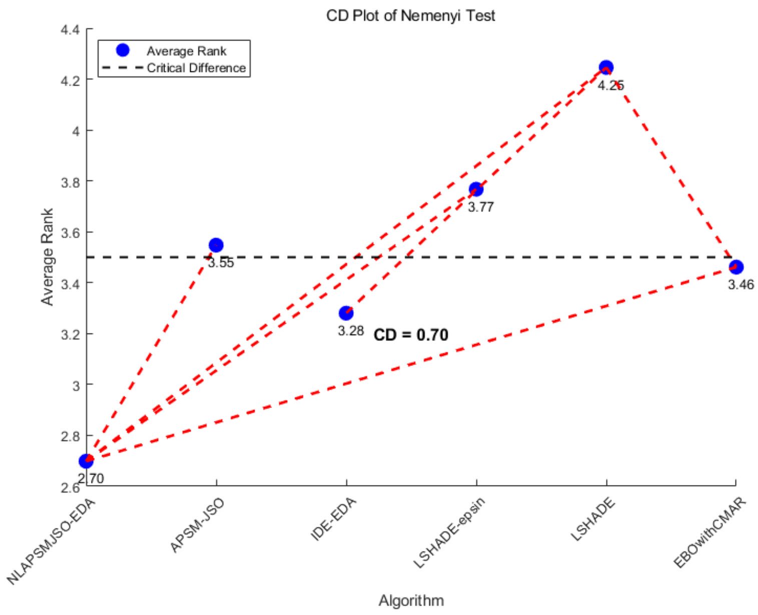

Figure 16. Comparing five DE-based rivals with NLAPSMjSO-EDA overall dimension-wise utilizing CDV.

4.3.3. Evaluation of Time Complexity

To evaluate the computational intricacy of NLAPSMjSO-EDA, this section uses the algorithm complexity assessment guidelines from the IEEE CEC 2017 benchmark [44]. The method complexity is $(T2 - T1)/T0$, where T0, T1, and T2 are specified in compliance with [44].

Table 15 contrasts six DE-based versions with NLAPSMjSO-EDA in terms of complexity. In Algorithm 2, T_0 is calculated by measuring the total execution time required to perform a set of basic mathematical operations repeated 1,000,000 times. The elapsed time serves as a benchmark for normalizing the computational complexity of other algorithms using the formula $(T2 - T1)/T_0$. Table 15 presents the computational complexity of NLAPSMjSO-EDA, APSM-jSO, IDE-EDA, LSHADE-Epsin, MadDE, LSHADE, and EBOwithCMAR for a more understandable comparison. Here, S represents seconds.

Table 15. Comparison of Time Complexity.

Algorithm	$\overline{T2}/S$				$(\overline{T2} - T1)/T0$			
	10D	**30D**	**50D**	**100D**	**10D**	**30D**	**50D**	**100D**
NLAPSMjSO-EDA	0.97188	1.6696	2.1975	5.0469	0.4389	0.5943	2.1698	1.632
APSM-jSO	0.57405	0.84655	1.2811	3.1345	2.0174	1.8693	2.1886	0.6933
IDE-EDA	0.65074	1.0083	1.5201	3.518	0.1018	1.4811	3.6839	2.3867
LSHADE-Epsin	5.1412	3.4443	3.0373	6.3919	2.1651	0.6745	0.882	1.7735
MadDE	1.1175	1.3686	4.7020	114.9105	14.1085	14.4811	15.4717	265.0896
LSHADE	0.78221	0.93036	1.3424	4.2259	11.0090	9.0066	9.2877	10.5189
EBOwithCMAR	1.8511	2.5996	4.0179	12.3922	0.4953	1.1651	3.2736	7.1415

T0 = 0.0212, T1(10D) = 0.96258, T1(30D) = 1.6822, T1(50D) = 2.2435, T1(100D) = 5.0815 for NLAPSMjSO-EDA; T0 = 0.0212, T1(10D) = 0.53128, T1(30D) = 0.88618, T1(50D) = 1.3275, T1(100D) = 3.1492 for APSM-jSO; T0 = 0.0212, T1(10D) = 0.6529, T1(30D) = 1.0397, T1(50D) = 1.4402, T1(100D) = 3.5686 for IDE-EDA; T0 = 0.0212, T1(10D) = 5.0953, T1(30D) = 3.4586, T1(50D) = 3.056, T1(100D) = 6.3543 for LSHADE-Epsin; T0 = 0.0212, T1(10D) = 1.4166, T1(30D) = 1.6756, T1(50D) = 5.03, T1(100D) = 120.5304 for MadDE; T0 = 0.0212, T1(10D) = 1.0156, T1(30D) = 1.1213, T1(50D) = 1.5393, T1(100D) = 4.4489 for LSHADE; T0 = 0.0212, T1(10D) = 1.8406, T1(30D) = 2.6243, T1(50D) = 3.9485, T1(100D) = 12.5436 for EBOwithCMAR.

NLAPSMjSO-EDA shows lower computational complexity than other algorithms in the 30D dimension.

At 10D, NLAPSMjSO-EDA is less complex than IDE-EDA and on par with EBOwithCMAR.

At 50D, NLAPSMjSO-EDA is less complex than LSHADE-Epsin and IDE-EDA, and comparable to APSM-jSO.

At 100D, NLAPSMjSO-EDA is less complex than APSM-jSO and comparable to LSHADE-Epsin.

Algorithm 2 Computer T0.

```
Input: Initialize $x$ to 0.55;
 1: Start timer;
 2: for $i$ from 1 to 1,000,000 do
 3:     $x = x + x$;
 4:     $x = x/2$;
 5:     $x = x \times x$;
 6:     $x = \sqrt{x}$;
 7:     $x = \log x$;
 8:     $x = \exp x$;
 9:     $x = x/(x + 2)$;
10: end for
11: Stop timer;
Output: Calculate $elapsed\_time$;
12: Display $elapsed\_time$;
```

4.3.4. Convergence and Robustness Analysis

This section discusses NLAPSMjSO-EDA's robustness and convergence performance. Only six benchmark functions in 50 dimensions—a unimodal function (F1), a basic multimodal function (F4), a hybrid function (F10), and three composite functions (F21, F23, and F30)—have convergence and box plots supplied for the sake of conciseness. The graphics show convergence and box plots for every function in every dimension.

Figure 17 shows the average convergence plots of NLAPSMjSO-EDA and 6 DE-based competitors over 51 independent runs on the mentioned functions. NLAPSMjSO-EDA demonstrates superior convergence speed compared to MadDE, APSM-jSO, IDE-EDA, LSHADE-Epsin, and LSHADE on the selected functions. On functions F10, F21, and F23, NLAPSMjSO-EDA's convergence speed is slightly inferior to that of EBOwithCMAR. Overall, the convergence performance of NLAPSMjSO-EDA is comparable to EBOwithCMAR, but it shows significant advantages over the other five competitors. These results indicate that NLAPSMjSO-EDA is a competitive choice among the evaluated algorithms.

The robustness of NLAPSMjSO-EDA is illustrated by the box plots of 51 separate run results, which are displayed in Figures 18–23. The center median, outliers, and first and third quartile values are displayed in every box graphic.The symbol "+" represents an outlier, which is commonly used in statistical analysis and data visualization to identify data points that significantly deviate from the main distribution. It can be observed that NLAPSMjSO-EDA exhibits outliers only in F21 and F23, similar to APSM-jSO. IDE-EDA shows three outliers, LSHADE-Epsin shows two outliers, MadDE shows three outliers, LSHADE shows three outliers, and EBOwithCMAR shows four outliers. This suggests that APSM-jSO's robustness is maintained even while its performance is improved.

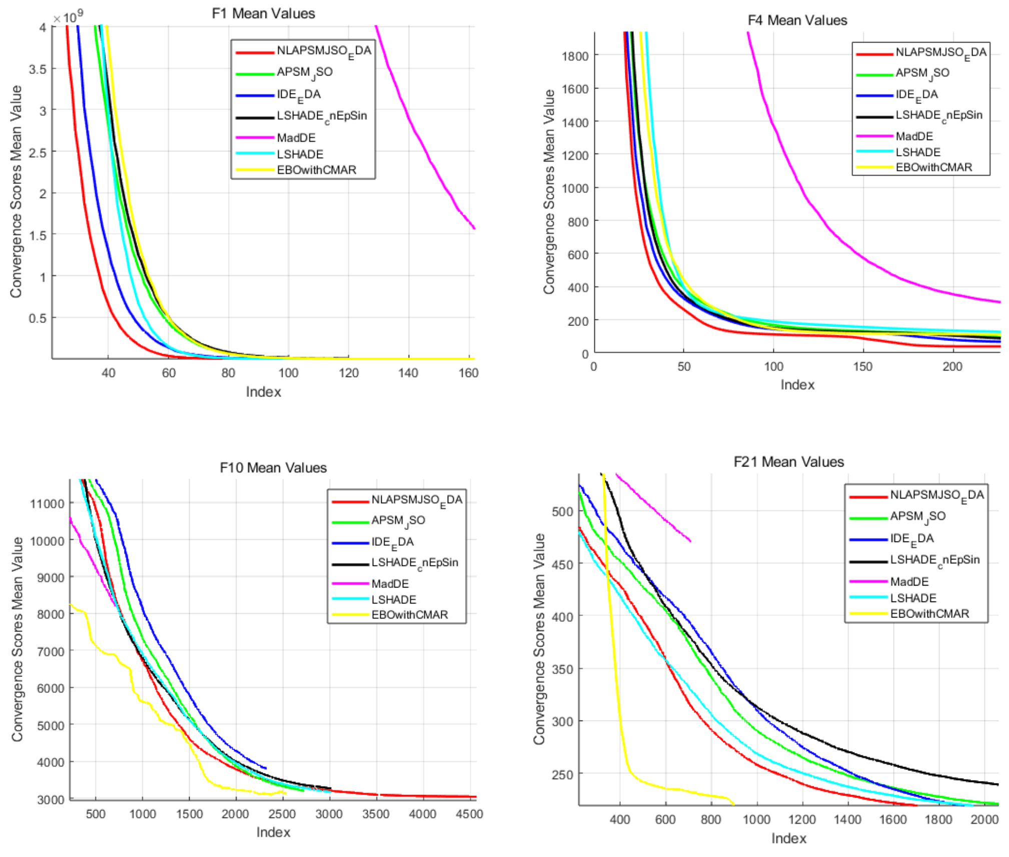

Figure 17. *Cont.*

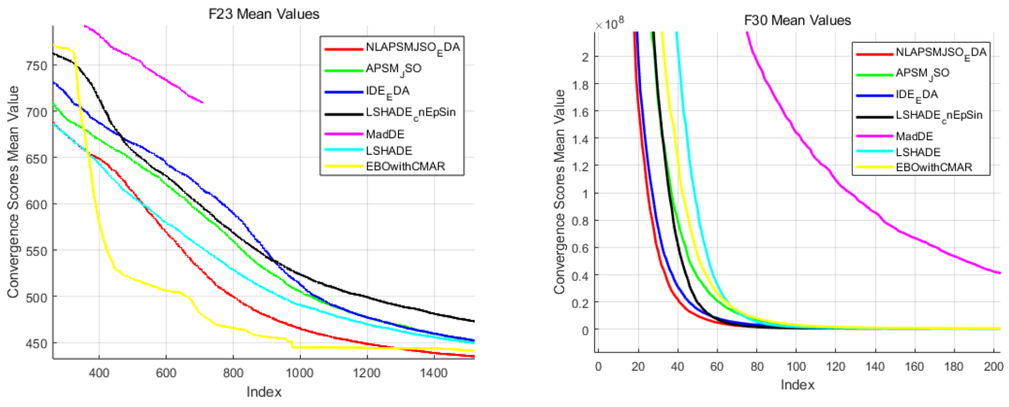

Figure 17. Convergence plots of results obtained by NLAPSMjSO-EDA and 6 competitors (D = 50).

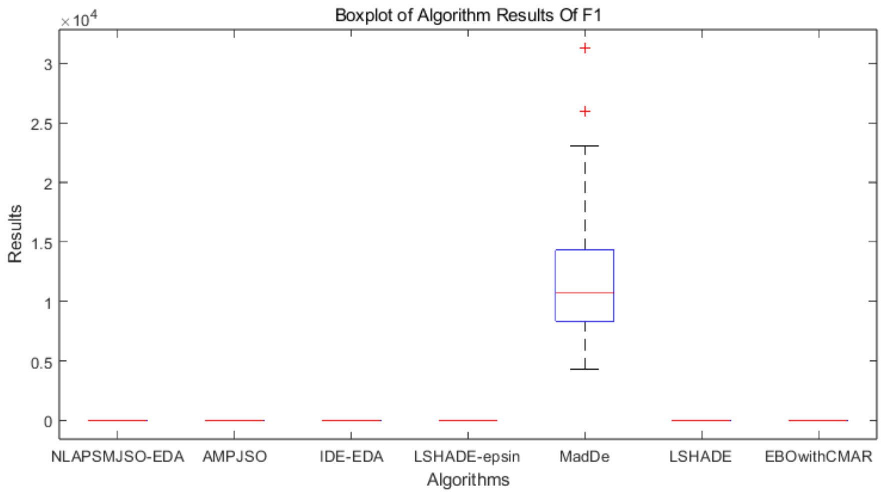

Figure 18. Boxplox of Algorithm Result of F1 (D = 50).

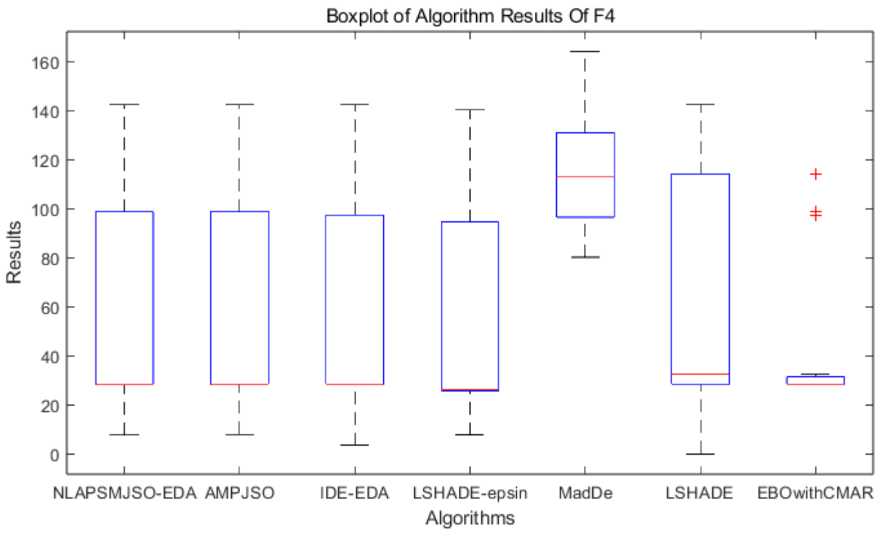

Figure 19. Boxplox of Algorithm Result of F4 (D = 50).

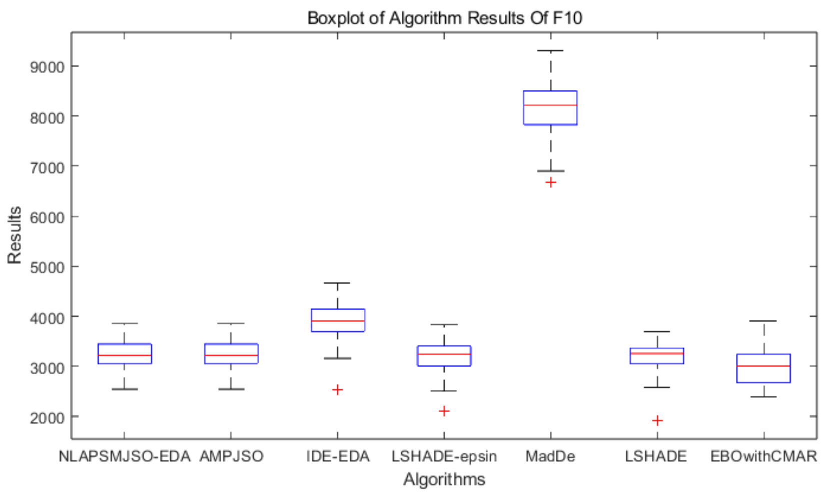

Figure 20. Boxplox of Algorithm Result of F10 (D = 50).

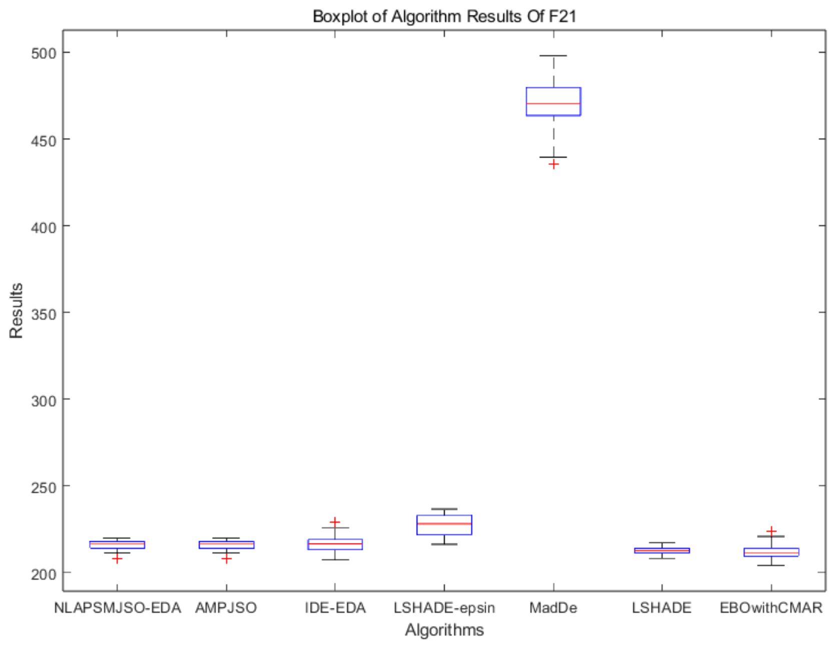

Figure 21. Boxplox of Algorithm Result of F21 (D = 50).

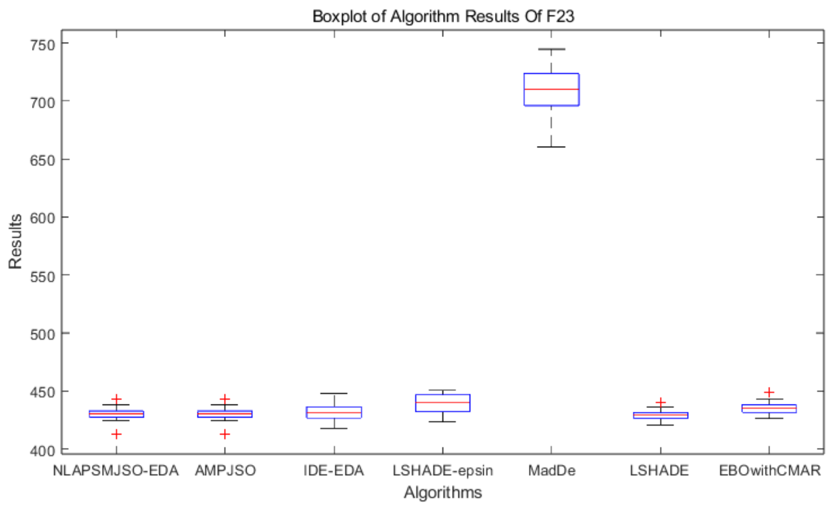

Figure 22. Boxplox of Algorithm Result of F23 (D = 50).

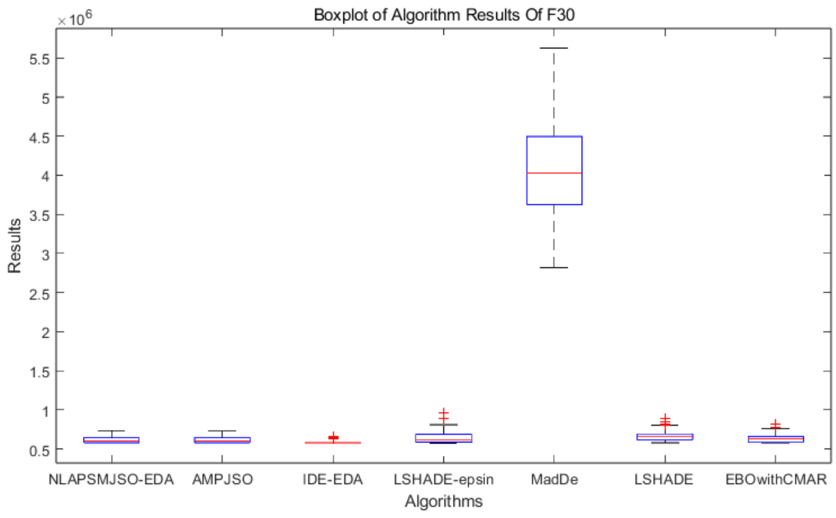

Figure 23. Boxplox of Algorithm Result of F30 (D = 50).

5. Conclusions and Future Work

In order to further boost APSM-jSO performance, this work suggests combining a new population reduction strategy (NL non-linear decrement) with a new population update technique (EDA Gaussian sampling to generate an elite population). By modifying the population reduction method, it is possible to explore the optimal solution for more generations, enhancing local exploration capabilities. Furthermore, by reducing the difficulty of escaping local optima brought on by the APSM-jSO algorithm's latter phases of diversity loss, the EDA algorithm can indirectly improve the system's capacity for global exploration. In an effort to significantly boost APSM-jSO efficiency, a novel population reduction approach (NL non-linear decrement) and a population updating method (EDA Gaussian sampling to produce an elite population) are introduced in this study. The findings of statistical analysis verify that in 10D, NLAPSMjSO-EDA performs similarly to IDE-EDA, is marginally worse than EBOwithCMAR and MadDE, and considerably surpasses APSM-jSO, LSHADE-Epsin, and LSHADE. In 30D, NLAPSMjSO-EDA significantly outperforms APSM-jSO, LSHADE-Epsin, LSHADE, and IDE-EDA, and is comparable to EBOwithCMAR. In 50D, NLAPSMjSO-EDA significantly outperforms all algorithms. In 100D, NLAPSMjSO-EDA significantly outperforms APSM-jSO, LSHADE-Epsin, and LSHADE, and is comparable to LSHADE-Epsin and IDE-EDA. Overall, in terms of average ranking across all dimensions, NLAPSMjSO-EDA ranks first and shows a significant performance gap compared to the other six algorithms.

In conclusion, the NLAPSMjSO-EDA proposed in this work is a possible jSO and EDA modification that may successfully further enhance the overall efficiency of APSM-jSO, even though APSM-jSO is already an extremely efficient method. To fully realize the performance potential of NLAPSMjSO-EDA, more adjustments to its parameters can be made in subsequent work. However, this study has some limitations that require further exploration. First, the applicability of the proposed method has primarily been validated through benchmark function tests, and its performance in real-world complex engineering problems needs further investigation and evaluation. Second, the algorithm has not yet been tested or optimized for multi-objective optimization problems, which represents an important direction for future research. The algorithm code has been open-sourced at https://github.com/ljxzzl/NLAPSMjSO-EDA (accessed on 16 January 2025).

Author Contributions: Conceptualization, H.K.; methodology, H.K.; project administration, J.L. and Y.S.; software, X.S.; validation, X.S. and Q.C.; visualization, J.L. and Q.C.; formal analysis, H.K.; investigation, Q.C.; resources, Y.S.; data curation, J.L.; writing—original draft preparation, J.L.; writing—review and editing, H.K. and X.S.; supervision: X.S.; funding acquisition: Y.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supplied by the National Natural Science Foundation of China (62366057) and Open Foundation of Key Laboratory of Software Engineering of Yunnan Province (Grant Nos. 2020SE308 and 2020SE309).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: We certify that we have no affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in this manuscript.

References

1. Vargas, D.E.C.; Lemonge, A.C.C.; Barbosa, H.J.C.; Bernardino, H.S. An interactive reference-point-based method for incorporating user preferences in multi-objective structural optimization problems. *Appl. Soft Comput.* **2024**, *165*, 112106. [CrossRef]
2. Brest, J.; Maucec, M.S.; Boskovic, B. Single objective real-parameter optimization: Algorithm jSO. In Proceedings of the 2017 IEEE Congress on Evolutionary Computation (CEC), Donostia, Spain, 5–8 June 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 1311–1318. [CrossRef]
3. Wang, X.; Tang, L. An adaptive multi-population differential evolution algorithm for continuous multi-objective optimization. *Inf. Sci.* **2016**, *348*, 124–141. [CrossRef]
4. Gong, W.; Cai, Z.; Liang, D. Adaptive ranking mutation operator based differential evolution for constrained optimization. *IEEE Trans. Cybern.* **2015**, *45*, 716–727. [CrossRef] [PubMed]
5. Sun, J.; Li, Y. Multi-feature fusion network for road scene semantic segmentation. *Comput. Electr. Eng.* **2021**, *92*, 107155. [CrossRef]
6. Wang, P.; Wang, D.; Zhang, X.; Li, X.; Peng, T.; Lu, H.; Tian, X. Numerical and experimental study on the maneuverability of an active propeller control-based wave glider. *Appl. Ocean Res.* **2020**, *104*, 102369. [CrossRef]
7. Chen, X. Novel dual-population adaptive differential evolution algorithm for large-scale multi-fuel economic dispatch with valve-point effects. *Energy* **2020**, *203*, 117874. [CrossRef]
8. Dong, Z.; Mao, S.; Perc, M.; Du, W.; Tang, Y. A distributed dynamic event-triggered algorithm with linear convergence rate for the economic dispatch problem. *IEEE Trans. Netw. Sci. Eng.* **2023**, *10*, 500–513. [CrossRef]
9. Jodlbauer, H.; Strasser, S. Capacity-driven production planning. *Comput. Ind.* **2019**, *113*, 103126. [CrossRef]
10. Mohamed, A.W.; Mohamed, A.K.; Elfeky, E.Z.; Saleh, M. Enhanced directed differential evolution algorithm for solving constrained engineering optimization problems. *Int. J. Appl. Metaheuristic Comput.* **2019**, *10*, 1–28. [CrossRef]
11. Shen, Y.; Liang, Z.; Kang, H.; Sun, X.; Chen, Q. A modified jSO algorithm for solving constrained engineering problems. *Symmetry* **2020**, *13*, 63. [CrossRef]
12. Lu, H.; Zhang, M.; Xu, X.; Li, Y.; Shen, H. Deep fuzzy hashing network for efficient image retrieval. *IEEE Trans. Fuzzy Syst.* **2020**, *29*, 166–176. [CrossRef]
13. Peng, J.; Li, Y.; Kang, H.; Shen, Y.; Sun, X.; Chen, Q. Impact of population topology on particle swarm optimization and its variants: An information propagation perspective. *Swarm Evol. Comput.* **2022**, *69*, 100990. [CrossRef]
14. Storn, R.; Price, K. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Optim.* **1997**, *11*, 341–359.:1008202821328. [CrossRef]
15. Price, K.; Storn, R.M.; Lampinen, J.A. *Differential Evolution: A Practical Approach to Global Optimization*; Springer Science & Business Media: New York, NY, USA, 2006.
16. Das, S.; Suganthan, P.N. Differential evolution: A survey of the state-of-the-art. *IEEE Trans. Evol. Comput.* **2010**, *15*, 4–31. [CrossRef]
17. Aslantas, V.; Kurban, R. Fusion of multi-focus images using differential evolution algorithm. *Expert Syst. Appl.* **2010**, *37*, 8861–8870. [CrossRef]
18. Segura, C.; Coello, C.A.C.; Hernández-Díaz, A.G. Improving the vector generation strategy of differential evolution for large-scale optimization. *Inf. Sci.* **2015**, *323*, 106–129. [CrossRef]

19. Mallipeddi, R.; Lee, M. An evolving surrogate model-based differential evolution algorithm. *Appl. Soft Comput.* **2015**, *34*, 770–787. [CrossRef]
20. Sun, X.; Wang, D.; Kang, H.; Shen, Y.; Chen, Q. A two-stage differential evolution algorithm with mutation strategy combination. *Symmetry* **2021**, *13*, 2163. [CrossRef]
21. Zhu, L.; Ma, Y.; Bai, Y. A self-adaptive multi-population differential evolution algorithm. *Nat. Comput.* **2020**, *19*, 211–235. [CrossRef]
22. Ma, Y.; Bai, Y. A multi-population differential evolution with best-random mutation strategy for large-scale global optimization. *Appl. Intell.* **2020**, *50*, 1510–1526. [CrossRef]
23. 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), Beijing, China, 6–11 July 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 1658–1665. [CrossRef]
24. Sun, X.; Jiang, L.; Shen, Y.; Kang, H.; Chen, Q. Success history-based adaptive differential evolution using turning-based mutation. *Mathematics* **2020**, *8*, 1565. [CrossRef]
25. Zhang, J.; Sanderson, A.C. JADE: Adaptive differential evolution with optional external archive. *IEEE Trans. Evol. Comput.* **2009**, *13*, 945–958. [CrossRef]
26. 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), Donostia, Spain, 5–8 June 2017; IEEE: Piscataway, NJ, USA, 2017. [CrossRef]
27. Tanabe, R.; Fukunaga, A. Success-history based parameter adaptation for differential evolution. In Proceedings of the 2013 IEEE Congress on Evolutionary Computation (CEC), Cancun, Mexico, 20–23 June 2013; IEEE: Piscataway, NJ, USA, 2013; pp. 71–78. [CrossRef]
28. 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, Spain, 5–8 June 2017; IEEE: Piscataway, NJ, USA, 2017. [CrossRef]
29. Biswas, S.; Saha, D.; De, S.; Cobb, A.D.; Das, A.; Jalaian, B.A. Improving differential evolution through Bayesian hyperparameter optimization. In Proceedings of the 2021 IEEE Congress on Evolutionary Computation (CEC), Kraków, Poland, 12–15 July 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 832–840. [CrossRef]
30. Li, Y.; Han, T.; Tang, S.;Huang, C.; Zhou, H.; Wang, Y. An improved differential evolution by hybridizing with estimation-of-distribution algorithm. *Inf. Sci.* **2023**, *619*, 439–456. [CrossRef]
31. Li, Y.; Zhang, W.; Li, X.; Ma, H. APSM-jSO: A novel jSO variant with an adaptive parameter selection mechanism and a new external archive updating mechanism. *Swarm Evol. Comput.* **2023**, *78*, 101283. [CrossRef]
32. 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; IEEE: Piscataway, NJ, USA, 2018. [CrossRef]
33. Xia, X.; Tong, L.; Zhang, Y.; Xu, X.; Yang, H.; Gui, L.; Li, Y.; Li, K. NFDDE: A novelty-hybrid-fitness driving differential evolution algorithm. *Inf. Sci.* **2021**, *579*, 33–54. [CrossRef]
34. 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), Kraków, Poland, 12–15 July 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 809–816. [CrossRef]
35. Elsayed, S.; Harnza, N.; Sarker, R. Testing united multi-operator evolutionary algorithms-II on single objective optimization problems. In Proceedings of the 2016 IEEE Congress on Evolutionary Computation (CEC), Vancouver, BC, Canada, 24–29 July 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 2966–2973. [CrossRef]
36. Mühlenbein, H.; Paaß, G. From recombination of genes to the estimation of distributions I. Binary parameters. In *Parallel Problem Solving from Nature*; Springer: Berlin, Germany, 1996; pp. 178–187.—982. [CrossRef]
37. Larrañaga, P.; Lozano, J.A. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*; Kluwer: Boston, MA, USA, 2002. . [CrossRef]
38. Pelikan, M.; Goldberg, D.E.; Lobo, F. A survey of optimization by building and using probabilistic models. *Comput. Optim. Appl.* **2002**, *21*, 5–20. https://10.1109/ACC.2000.879173. [CrossRef]
39. Zhou, B.; Huang, Y. An adaptive archive differential evolution with non-linear population size reduction and selective pressure. *Inf. Sci.* **2024**, *682*, 121273. [CrossRef]
40. Zhou, B.-H.; Hu, L.-M.; Zhong, Z.-Y. A hybrid differential evolution algorithm with estimation of distribution algorithm for reentrant hybrid flow shop scheduling problem. *Neural Comput. Appl.* **2018**, *30*, 193–209. [CrossRef]
41. Du, K.-L.; Swamy, M.N.S. Estimation of distribution algorithms. *Search and Optimization by Metaheuristics: Techniques and Algorithms Inspired by Nature*; Springer International Publishing: Cham, Switzerland, 2016; pp. 105–119. [CrossRef]

42. Mohamed, A.W.; Hadi, A.A.; Mohamed, A.K.; Awad, N.H. Evaluating the performance of adaptive gaining-sharing knowledge based algorithm on CEC2020 benchmark problems. In Proceedings of the 2020 IEEE Congress on Evolutionary Computation (CEC), Glasgow, UK, 19–24 July 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 1–8. [CrossRef]
43. Ren, Z.; Liang, Y.; Wang, L.; Yao, X. Anisotropic adaptive variance scaling for Gaussian estimation of distribution algorithm. *Knowl. Based Syst.* **2018**, *146*, 142–151. [CrossRef]
44. Awad, N.H.; Ali, M.Z.; Liang, J.J.; Qu, B.Y.; Suganthan, P.N. *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.
45. Li, Y.; Han, T.; Zhou, H.; Zhao, X. An adaptive L-SHADE algorithm and its application in UAV swarm resource configuration problem. *Inf. Sci.* **2022**, *606*, 350–367. [CrossRef]
46. Yi, W.; Chen, Y.; Pei, Z.; Lu, J. Adaptive differential evolution with ensembling operators for continuous optimization problems. *Swarm Evol. Comput.* **2022**, *69*, 100994. [CrossRef]
47. Pluhacek, M.; Viktorin, A.; Kadavy, T.; Kazikova, A. On the common population diversity measures in metaheuristics and their limitations. In Proceedings of the 2021 IEEE Symposium Series on Computational Intelligence (SSCI), Orlando, FL, USA, 7–10 December 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 1–7. [CrossRef]