

Cite this paper:

Brest, J., Popič, J., Pšeničnik, B. et al. Population size reduction methods in differential evolution: a comprehensive review and experimental analysis. *Artif Intell Rev* (2026). <https://doi.org/10.1007/s10462-026-11712-5>

Population Size Reduction Methods in Differential Evolution: A Comprehensive Review and Experimental Analysis

Janez Brest^{1*}, Jan Popič¹, Blaž Pšeničnik¹, Borko Bošković¹,
Mirjam Sepesy Maučec²

¹Institute of Computer Science, Faculty of Electrical Engineering and
Computer Science, University of Maribor, Koroška c. 46, Maribor, 2000,
Slovenia.

²Institute of Electronics and Telecommunications, Faculty of Electrical
Engineering and Computer Science, University of Maribor,
Koroška c. 46, Maribor, 2000, Slovenia.

*Corresponding author(s). E-mail(s): janez.brest@um.si;
Contributing authors: jan.popic1@um.si; blaz.psenicnik1@um.si;
borko.boskovic@um.si; mirjam.sepesy@um.si;

Abstract

Optimization plays a crucial role in modern engineering and science. Evolutionary computing techniques are a branch of artificial intelligence (AI) and have attracted a lot of attention recently. Scientists, engineers, and practitioners need to understand how evolutionary computing techniques work and be fluent in using AI-based optimization techniques to solve engineering optimization problems. The Differential Evolution (DE) algorithm, which is classified as a population-based evolutionary algorithm, is primarily intended for solving complex optimization problems. The original DE algorithm uses a fixed population size, whereas modern DE variants adopt advanced strategies for reducing the population size over the evolution process. In this review article, 204 studies are included and we offer not only an overview of the population size reduction methods, but also a comparative analysis of selected methods by integrating them into the state-of-the-art algorithm and performing experimental work on the CEC 2024 benchmark functions and two real-world problems. Benchmarking demonstrates that the choice of population size adjustment strategy affects optimization performance, with its effectiveness depending on the problem characteristics and dimensionality. According to the Friedman test applied to 15

methods, 10 non-linear population size reduction methods were ranked higher than the linear population size reduction method on 100-dimensional problems. Moreover, a non-linear population size reduction method achieved the top rank on both real-world optimization problems. However, no single population size adjustment strategy consistently dominates across all problem dimensions and function families.

Keywords: Differential Evolution, Numerical optimization, Population size adjustment, Linear population size reduction, Nonlinear population size reduction

1 Introduction

Differential Evolution (DE) emerged as a very competitive evolutionary algorithm more than three decades ago (Storn and Price 1997). Since then, DE, in its original form as well as its variants, has been applied successfully to solve numerous real-world optimization problems (Das and Suganthan 2011; Brest et al. 2006; Neri and Tirronen 2010; Das et al. 2016).

DE is a population-based stochastic search algorithm that imitates natural evolutionary processes such as mutation, crossover, and selection. The key parameters of DE are scaling factor (F), crossover rate (CR), and population size (NP). Choosing their appropriate values is crucial for achieving good performance. Early DE variants relied on constant parameter values, but subsequent studies have demonstrated that it is beneficial to adapt them throughout the evolutionary process. Based on that observation, many powerful and efficient variants of DE have been proposed in the literature. The algorithms were systematically reviewed in various papers, with the surveys conducted in different directions. Tables 1 and 2 collect DE reviews and surveys. In (Al-Dabbagh et al. 2018), the authors proposed an extended taxonomy for evolutionary algorithm parameter settings, covering the different types of parameter configurations used in evolutionary computation. Based on this taxonomy, they introduced a general classification of adaptive DE algorithms that categorizes them according to their parameter-control mechanisms and the number of DE strategies employed in the implementation. Ye et al. (2025) surveyed adaptation mechanisms that span many different directions, ranging from control parameters to learning schemes. Reyes-Davila et al. (2025) analyzed the variants of DE operators with focus on mutation schemes, crossover, and selection operators. Pan et al. (2026) surveyed DE algorithms employing exponential crossover. The authors analyzed their behavioral properties in comparison with binomial crossover. Piotrowski et al. (2026) presented a comprehensive experimental survey of SHADE- and L-SHADE-based adaptive DE algorithms.

Although several reviews of DE have been published, they primarily focus on the adaptation of the F and CR parameters and on the general development of DE and its applications across various optimization domains. This review paper focuses on a single parameter: population size, which is typically assigned a fixed value based on empirical evidence. Regarding how DE’s mutation and crossover work, it is clear

Table 1: DE surveys and reviews.

Reference	Title and note
Neri and Tirronen (2010)	<i>Recent advances in differential evolution: a survey and experimental analysis</i> Surveys DE and its recent advances. Classifies the DE modifications into two groups: (1) algorithms that integrate additional components within the DE structure; (2) algorithms that employ a modified DE structure.
Das and Suganthan (2011)	<i>Differential evolution: A survey of the state-of-the-art</i> Surveys state-of-the-art DE algorithms and their real-world applications; and analyzes DE variants based on control parameters and mutation schemes.
Dragoi and Dafinescu (2016)	<i>Parameter control and hybridization techniques in differential evolution: a survey.</i> Surveys DE in two main directions: (1) the replacement of manual control parameter setting with adaptive and self-adaptive methods; (2) hybridization with other algorithms. Presents approaches of mixing DE with global search algorithms, DE with local search algorithms, and DE with global and local search algorithms.
Das et al. (2016)	<i>Recent advances in differential evolution – an updated survey</i> Surveys DE algorithms, parameter adaptation in DE, applications, and hybridization of DE and other methodologies.
Piotrowski (2017)	<i>Review of differential evolution population size</i> Briefly reviews the opinions regarding DE population size setting and verifies the impact of the population size on the performance of DE algorithms. Conducts experiments with ten DE algorithms with fixed population size and four DE algorithms with flexible population size on CEC 2005 benchmarks and CEC 2011 real-world problems.
Al-Dabbagh et al. (2018)	<i>Algorithmic design issues in adaptive differential evolution schemes: Review and taxonomy</i> Provides the extended taxonomy of parameter settings in evolutionary algorithms and proposes two taxonomies of parameters in DE algorithms. Investigates the various algorithmic design schemes that have been used in the different variants of adaptive DE and conveys them in a new classification style. Provides a comparative analysis of the algorithmic schemes and offers several guidelines for designing and implementing adaptive DE algorithms.

Table 2: DE surveys and reviews.

Reference	Title and note
Tanabe and Fukunaga (2019)	<i>Reviewing and benchmarking parameter control methods in differential evolution</i> Reviews 24 parameter control methods for scale factor and crossover rate in DE variants. Conducts a benchmarking study of these methods and analyzes their performance by comparing them with recent algorithms.
Bilal et al. (2020)	<i>Differential Evolution: A review of more than two decades of research</i> Surveys nearly 25 years of existence of DE, covering 283 research articles with a focus on DE operations, parameters, and hybridized variants, along with various successful applications of DE.
Reyes-Davila et al. (2025)	<i>Differential evolution: a survey on their operators and variants</i> Surveys the variants of DE operators and their impact on the evolutionary computation literature over the years, and provides an analysis of each year’s trends, the improvements that marked a milestone in the DE research, and the feasible future directions of the algorithm.
Ye et al. (2025)	<i>A comprehensive survey of adaptive strategies in differential evolutionary algorithms</i> Surveys the DE algorithms based on adaptive strategies and provides a new taxonomy of six adaptive strategies. Highlights the advantages and disadvantages of adaptive strategies and highlights promising directions of categorized adaptive strategies.
Pan et al. (2026)	<i>Differential evolution with exponential crossover: A survey.</i> Surveys DE algorithms that employ exponential crossover. A three-category classification framework is proposed to organize existing algorithms and guide future research. The paper analyzes how the crossover rate affects inherited segment lengths in exponential vs. binomial crossover and reviews real-world applications, as well as future challenges such as parameter control and framework integration.
Piotrowski et al. (2026)	<i>A comprehensive experimental survey of L-SHADE and SHADE-based DE algorithms</i> Experimentally surveys SHADE and L-SHADE-based adaptive differential evolution algorithms, providing comprehensive results across different problem dimensionalities, difficulty levels, and both real-world and numerical benchmark functions. The study provides practical guidance for selecting suitable SHADE/L-SHADE variants depending on problem characteristics and application scenarios and identifies promising directions for future research.

that a larger population generally leads to greater diversity among individuals. In contrast, when the population size is too small, the algorithm tends to lose diversity prematurely, making it more susceptible to stagnation or early convergence.

To the best of our knowledge, the population size has not been systematically examined in many review articles. [Piotrowski \(2017\)](#) briefly reviews the opinions regarding DE population size setting and verifies the impact of the population size on the performance of DE algorithms. Using a too small population size limits the number of available moves, which may lead to stagnation or premature convergence. On the other hand, a very large population may waste many function evaluations with almost no exploratory moves. [Piotrowski \(2017\)](#) presented four ways on how the population size of DE algorithms could be set: (1) setting population size for each problem separately, without any motivation or according to the 'expert knowledge', and keeping it fixed for the whole run; (2) relating population size to the dimensionality of the problem D , and keeping it fixed for the whole run; (3) setting the same fixed population size for all problems, irrespective of their dimensionality or properties, (4) allowing population to vary during run, usually according to specified adaptation rules.

Several methods were proposed for adjusting the population size throughout the evolutionary process. They follow the idea that having a large population with increased diversity at the beginning enhances the algorithm's ability to explore the unknown search space. However, as optimization progresses, diversity must gradually diminish to enable convergence toward one of the optimal solutions. Since the population size is larger in the early stages and smaller in the later stages of the evolutionary process, an overall decrease occurs, even if minor increases appear in between. Thus, the population size may decline monotonically or fluctuate during the evolutionary process while still following a general downward trend. Adaptive population size is highly recommended, especially for higher-dimensional and real-world problems. However, which specific algorithms with population size adaptation perform better depends on the number of allowed function calls. Experimental evidence showed that algorithms with variable population size are never among the worst ([Piotrowski et al. 2017](#)). Although variable population sizes do not guarantee good performance, they help avoid pitfalls. In ([Piotrowski et al. 2017](#)) it was also found that five out of six tested algorithms with variable population sizes were not ranked among the worst methods according to averaged rankings across 22 problems in any of the six competitions performed. In another study ([Saad et al. 2024](#)), the authors concluded that using a population size based on problem dimension or a fixed value cannot be considered an optimal approach. In ([Ye et al. 2025](#)), population size adjustment was briefly reviewed alongside other adaptation strategies. The authors highlight the considerable potential for incorporating adaptive strategies into various components of DE algorithms, including the population size. In ([Piotrowski et al. 2026](#)), population size reduction was addressed because it is integrated into several reviewed L-SHADE-based algorithms.

Despite the growing importance of population-size adaptation in differential evolution, no comprehensive review has systematically analyzed the large number of population-size adjustment mechanisms proposed during the last decade. Research surveys in DE have predominantly addressed parameter control, operator design, and

algorithmic enhancements rather than population size adaptation. The only dedicated review of the DE population size was published in (Piotrowski 2017) and therefore does not cover the substantial body of work developed afterward. While that study provided important insights into population-size setting and adaptation, it covered methods available up to 2017. Since then, numerous new population-size adaptation mechanisms have been proposed. Nevertheless, no study has systematically compared newer representative population-size adjustment mechanisms by integrating them into a common state-of-the-art DE framework and using recent benchmark suites. Consequently, the current understanding of similarities, differences, and practical effectiveness of modern population-size adaptation strategies remained fragmented.

To fill this gap, we present a review focusing on recent developments in differential evolution, with particular emphasis on population size adjustment strategies. The contributions of this paper are:

- A comprehensive review of population size adjustment methods in DE published between 2017 and 2026 (in 2026 from January to May inclusive).
- Unified classification of population-size adjustment methods into two groups: linear and non-linear population size adjustments.
- Comparative analysis of common design principles of population-size adjustment methods.
- Controlled empirical evaluation of selective population size adjustment methods within a state-of-the-art DE-based algorithm.
- Validation on the CEC 2024 benchmark suite with problem dimensions $D = 10, 30, 50$, and 100 , and two real-world optimization problems.

The algorithms reviewed employ varying notations in their original descriptions; therefore, a unified notation is adopted here for consistency. Table 3 summarizes the symbols and their brief descriptions that are used throughout the review of the algorithms presented in this paper.

The rest of the paper is organized as follows. Section 2 introduces the classical DE and jSO algorithms and provides a concise overview of population size reduction methods developed before 2017. A review methodology is given in Section 3. Section 4 presents the population size adjustment methods developed for DE-based algorithms since 2017. Common properties of these methods are summarized in Section 5. Section 6 reports simulation results obtained from the comparison of selected methods, while Section 7 presents experimental results on real-world optimization problems. Section 8 is dedicated to discussion. Finally, Section 9 summarizes the paper and gives directions for future work.

2 Background

In this section, we give an overview of the classic DE algorithm, the jSO (Brest et al. 2017) algorithm and of selected DE-based algorithms with population size adjustment proposed before 2017. The jSO algorithm is a strong and widely used SHADE-family baseline algorithm presented at CEC 2017 Competition on Single objective real-parameter optimization (Awad et al. 2016). We use the jSO algorithm as the

Table 3: Symbols and notation used in this paper.

Symbol	Description
D	problem size
$\mathbf{x}$	solution vector, $\mathbf{x} = (x_1, x_2, \dots, x_D)$
$\mathbf{x}_{best}$	the best solution vector
$\mathbf{x}_{pBest}$	top p -best solution vector
G	maximum number of generations
g	generation index, $g \in \{0, 1, \dots, G\}$
$\mathbf{P}_g$	population in generation g ($\mathbf{P}_0$ is the initial population)
NP	population size
NP_{init}	initial population size
NP_{end}	final population size
NP_{min}	minimum value of the population size during the optimization process
NP_{max}	maximum value of the population size during the optimization process
CR	crossover rate
F	scale factor
F_w	weighted scale factor
$MaxFEs$	maximum number of function evaluations
$nfes$	number of function evaluations, $nfes \in \{1, 2, \dots, MaxFEs\}$
f	fitness function value, $f = f(\mathbf{x})$
f_{best}	best fitness function value
f_{worst}	worst fitness function value
$diffPS$	number of different population sizes during the optimization process
p_{max}	initial value of parameter p
p_{min}	final value of parameter p
$\log$	natural logarithm
SR	success rate

baseline, into which we integrate various population-size adjustment strategies originally proposed in different DE variants. Such an implementation enables more explicit comparisons and the assessment of each strategy’s impact, as other algorithm-specific mechanisms do not confound its performance.

The jSO and other DE-based algorithms with population size adjustment proposed before 2017 laid the groundwork for many subsequent developments in population adaptation, discussed in Sections 4 and 5.

2.1 Differential Evolution

In this subsection, we briefly review the DE algorithm (Storn and Price 1997; Price et al. 2005), as it serves as the basic algorithm for the studies of more advanced algorithms and their methods in this paper.

A goal of an optimization algorithm is to find a solution vector $\mathbf{x} = (x_1, x_2, \dots, x_D)$ such that optimizes $f(\mathbf{x})$. D is the dimensionality of the function f . The domains of the components of the vector $\mathbf{x}$ are defined by their lower and upper bounds: $x_{j,low}, x_{j,upp}$; $j \in \{1, 2, \dots, D\}$.

DE belongs to the group of population-based evolutionary algorithms, as it has a population of NP individuals or vectors. In generation g , the population $\mathbf{P}_g$ can be

Table 4: The mutation strategies in the classic DE algorithm.

Name	Mutation strategy
DE/rand/1	$\mathbf{v}_{i,g} = \mathbf{x}_{r_1,g} + F(\mathbf{x}_{r_2,g} - \mathbf{x}_{r_3,g})$
DE/best/1	$\mathbf{v}_{i,g} = \mathbf{x}_{best,g} + F(\mathbf{x}_{r_1,g} - \mathbf{x}_{r_2,g})$
DE/current-to-best/1	$\mathbf{v}_{i,g} = \mathbf{x}_{i,g} + F(\mathbf{x}_{best,g} - \mathbf{x}_{i,g}) + F(\mathbf{x}_{r_1,g} - \mathbf{x}_{r_2,g})$
DE/best/2	$\mathbf{v}_{i,g} = \mathbf{x}_{best,g} + F(\mathbf{x}_{r_1,g} - \mathbf{x}_{r_2,g}) + F(\mathbf{x}_{r_3,g} - \mathbf{x}_{r_4,g})$
DE/rand/2	$\mathbf{v}_{i,g} = \mathbf{x}_{r_1,g} + F(\mathbf{x}_{r_2,g} - \mathbf{x}_{r_3,g}) + F(\mathbf{x}_{r_4,g} - \mathbf{x}_{r_5,g})$

represented by:

$$\mathbf{P}_g = (\mathbf{x}_{1,g}, \dots, \mathbf{x}_{i,g}, \dots, \mathbf{x}_{NP,g}), \quad (1)$$

where $g \in \{0, 1, \dots, G\}$ is a generation index, and G is the maximum number of generations. Each vector consists of D variables:

$$\mathbf{x}_{i,g} = (x_{i,1,g}, x_{i,2,g}, \dots, x_{i,D,g}). \quad (2)$$

In many optimization problems, variables are restricted by lower and upper bounds arising from physical limitations or practical considerations. These constraints are commonly known as bound constraints or box constraints (Biedrzycki et al. 2019). The initial population, $\mathbf{P}_0$, is usually selected uniformly at random within the specified lower and upper bounds defined for each individual's variable $x_{j,i}$:

$$x_{j,i} = x_{j,low} + rand_j(0, 1) \times (x_{j,upp} - x_{j,low}) \quad \text{for } j \in \{1, 2, \dots, D\}, \quad (3)$$

where $rand(0, 1)$ is a random number between 0 and 1. Instead of random population initialization, a different initialization can be used that involves heuristic knowledge of the optimization problem.

After initialization, the population of NP vectors is evolved throughout G generations, leading the vectors in the search space towards the global optimum. At the end of the evolutionary process, DE stops and returns the best-fitting vector and its fitness value as a final solution. During each generation, the DE algorithm repeats three operations for each individual. These operations are mutation, crossover, and selection.

The performance of the DE algorithm is largely dependent on the mutation process (Zhang et al. 2023a). In the mutation, the population size greatly affects the selection of the base vector and the difference vector. A large population size can enrich the diversity of individuals, while a small population size may produce few mutant individuals and cause the algorithm to fall into the local optima, although the convergence speed is faster.

2.1.1 Mutation

During the mutation operation, DE creates a mutant vector $\mathbf{v}_{i,g}$. Table 4 shows the mutation strategies that have been proposed for the classic DE algorithm. The indexes

r_1-r_5 represent the random and mutually different integers generated within the range $\{1, \dots, NP\}$ and also different from index i . $\mathbf{x}_{best,g}$ is the best vector in generation g .

Each strategy has a different ability to maintain population diversity, which might influence the algorithm's convergence rate during the optimization process. For example, the DE/rand/1 mutation strategy has better exploration ability while the DE/best/1 mutation strategy has better exploitation ability. A mutant vector $\mathbf{v}_{i,g}$ is calculated as the sum of a base vector and a scaled difference of two other vectors. The scaled difference can be repeated twice, for example, in DE/rand/2, or even multiple times. Several other DE mutation strategies are known, and the reader is referred to the DE surveys (Das and Suganthan 2011; Neri and Tirronen 2010; Das et al. 2016; Al-Dabbagh et al. 2018; Ye et al. 2025).

2.1.2 Crossover

The next operation that follows the mutation operation is called crossover. Classical DE has already introduced two types of crossing operations, namely binomial and exponential (Das and Suganthan 2011). The DE/rand/1/bin notation indicates the selected mutation strategy followed by binomial crossover. We encounter binomial crossover frequently in practice, so we will describe it in detail.

Two vectors, the mutant vector $\mathbf{v}_{i,g}$ and the parent vector $\mathbf{x}_{i,g}$, are used in binomial crossover in order to create a trial vector $\mathbf{u}_{i,g}$ as follows:

$$u_{j,i,g} = \begin{cases} v_{j,i,g}, & \text{if } (rand_j(0,1) \leq CR) \text{ or } (j = j_{rand}), \\ x_{j,i,g}, & \text{otherwise,} \end{cases} \quad (4)$$

for $j = 1, 2, \dots, D$ and $i = 1, 2, \dots, NP$. $rand_j(0,1)$ is the j -th evaluation of the uniform random generator number. The crossover parameter $CR \in [0,1]$ presents the probability of generating components for the trial vector from the mutant vector. If a component was not selected from the mutant vector, then it is selected from the parent vector. The randomly chosen index $j_{rand} \in \{1, 2, \dots, D\}$ is responsible for the trial vector containing at least one component from the mutant vector.

Once the crossover operation is complete, the bounds of the trial vector components are verified. If any component from the trial vector $\mathbf{u}_{i,g}$ is out of bounds, a repair mechanism is applied to move the individual back inside the search space bounds. The mentioned components are usually reinitialized randomly; however, alternative repair mechanisms can also be used.

2.1.3 Selection

The last operation is called selection. Before the selection takes place, the trial vector $\mathbf{u}_{i,g}$ is evaluated so that the objective function $f(\mathbf{u}_{i,g})$ is calculated.

During the selection operation, two vectors, the trial vector $\mathbf{u}_{i,g}$ and its corresponding parent vector $\mathbf{x}_{i,g}$, are compared based on their objective function values. The better one will take place as a member of the next generation, while the other one is discarded. The selection operation, under the assumption that the optimization

problem is a minimization problem, is defined as follows:

$$\mathbf{x}_{i,g+1} = \begin{cases} \mathbf{u}_{i,g}, & \text{if } f(\mathbf{u}_{i,g}) \leq f(\mathbf{x}_{i,g}), \\ \mathbf{x}_{i,g}, & \text{otherwise.} \end{cases} \quad (5)$$

This greedy selection operation is common in DE, but it is rarely used in other evolutionary algorithms, where tournament, roulette wheel, and other selection methods are more common.

2.2 The jSO algorithm

The jSO (Brest et al. 2017) algorithm, which achieved 2nd place in the CEC 2017 (Awad et al. 2016) competition, is a revised version of the iL-SHADE (Brest et al. 2016) algorithm proposed by the same authors. To improve the performance of the algorithm, the authors introduced a new weighted mutation strategy and several modifications of the control parameters. The novel mutation strategy, called DE/current-to- p Best-w/1, incorporates a new inertia weight F_w , as follows:

$$\mathbf{v}_{i,g} = \mathbf{x}_{i,g} + F_w(\mathbf{x}_{pBest,g} - \mathbf{x}_{i,g}) + F(\mathbf{x}_{r_1,g} - \mathbf{x}_{r_2,g}). \quad (6)$$

Vector $\mathbf{x}_{pBest,g}$ is randomly selected from top $100p\%$ best individuals in the current population. The proposed weighted mutation strategy uses a smaller factor F_w when the difference vectors include $\mathbf{x}_{pBest,g}$ during the early stages of evolution and gradually increases F_w as the evolutionary process progresses:

$$F_w = \begin{cases} 0.7 \times F, & \text{if } nfes < 0.2 \times MaxFEs, \\ 0.8 \times F, & \text{if } nfes < 0.4 \times MaxFEs, \\ 1.2 \times F, & \text{otherwise,} \end{cases} \quad (7)$$

where $nfes$ and $MaxFEs$ are the current and maximum number of evaluations of the objective function, respectively. This effectively reduces the influence of the vector $\mathbf{x}_{pBest,g}$ in the earlier stages.

Furthermore, the generated control parameters F and CR are also bounded according to the current evolutionary stage:

$$F = \begin{cases} 0.7, & \text{if } F > 0.7 \wedge nfes < 0.6 \times MaxFEs, \\ F, & \text{otherwise.} \end{cases} \quad (8)$$

$$CR = \begin{cases} 0.7, & \text{if } CR < 0.7 \wedge nfes < 0.25 \times MaxFEs, \\ 0.6, & \text{if } CR < 0.6 \wedge 0.25 \times MaxFEs \leq nfes < 0.5 \times MaxFEs, \\ CR, & \text{otherwise.} \end{cases} \quad (9)$$

The linear population size reduction (LPSR) mechanism, as well as the use of an external archive, remains unchanged. Therefore, jSO uses the LPSR proposed

by Tanabe and Fukunaga (2014) in L-SHADE. It is a special case of simple variable population size (Laredo et al. 2009). In each generation, the population size is calculated as:

$$NP_{g+1} = \text{round} \left[\left(\frac{NP_{end} - NP_{init}}{MaxFEs} \right) \times n_{fes} + NP_{init} \right]. \quad (10)$$

Population size reduction mechanism may effectively save computing resources during the evolution process and speed up convergence. The parameter settings in the jSO algorithm are: $NP_{init} = 25 \times \log(D)\sqrt{D}$, $NP_{end} = 4$, $\mu_F = 0.3$, $\mu_{CR} = 0.8$, $F \sim \mathcal{C}(\mu_F, 0.1)$, $CR \sim \mathcal{N}(\mu_{CR}, 0.1)$, $r^{arc} = 1.6$, $p_{max} = 0.25$, $p_{min} = p_{max}/2$, and $H = 5$; $\mathcal{C}$ and $\mathcal{N}$ denote Cauchy and Normal distribution respectively and $\log$ denotes the natural logarithm.

Later, in the experimental part of this review, we use the jSO algorithm as the baseline, into which we integrate various population-size adjustment strategies originally proposed in different DE variants. Such an implementation enables more explicit comparisons and the assessment of each strategy’s impact, as other algorithm-specific mechanisms do not confound its performance.

2.3 Selected DE-based algorithms with population size adjustment published before 2017

In this subsection, we present some DE-based algorithms with population size adjustment proposed before 2017, as they laid the groundwork for many subsequent developments in population adaptation, and we refer to them in this review paper.

2.3.1 DESAP (Teo 2006)

Teo (2006) introduced the Differential Evolution with Self-Adapting Populations (DESAP) algorithm. The proposed algorithm has two variants in which the population size is automatically determined at initialization and subsequently adapted throughout the evolutionary process. Each individual is associated with a self-adaptive, population-size-related parameter π , which is modified through crossover and mutation. The two variants, named DESAP-Abs and DESAP-Rel, differ only in their encoding of the population size. The initial population size is calculated as follows:

$$NP_{init} = 10 \times D. \quad (11)$$

For DESAP-Abs, the population size of the next generation is calculated as the average of the population-size attributes π_i over all individuals, where π_i represents an absolute population size preference:

$$NP_{g+1} = \text{round} \left(\sum_{i=1}^{NP_g} \pi_i / NP_g \right). \quad (12)$$

The parameter π_i for each individual is initialized as $\pi_i = NP_{init} + \mathcal{N}(0, 1)$. During the mutation operation, it is updated as $\pi_i \leftarrow \pi_i + \text{int}(\mathcal{N}(0.5, 1))$.

In DESAP-Rel, each π represents a relative growth rate and is initialized uniformly in the interval $[-0.5, 0.5]$. The population size of the next generation is then computed as:

$$NP_{g+1} = \text{round}(NP_g + (\pi \times NP_g)). \quad (13)$$

Elitism and culling are inherently incorporated in both versions through population size transitions between generations. When the population size increases, elitism is applied by retaining all individuals from the current generation and duplicating the best-performing individual to fill the additional slots. Conversely, when the population size decreases, culling occurs such that only the top NP_{g+1} individuals are preserved, while the remaining individuals are eliminated.

2.3.2 dynNP-DE (Brest and Sepesy Mauřec 2008)

To effectively balance the exploration–exploitation trade-off, a dynNP-DE algorithm was introduced, incorporating a dynamically adaptive population size, denoted as dyn-NP (Brest and Sepesy Mauřec 2008). At the beginning of the evolutionary process, a larger population facilitates broader exploration of the search space, while the population size is progressively reduced as the process advances. In the final stage, the algorithm intensifies its focus on refining and optimizing the best-performing individuals identified thus far.

If $diffPS$ denotes the number of different population sizes, then $diffPS - 1$ reductions need to be performed. $NP_1 = NP_{init}$ is the initial population size and NP_p (where $p = 1, 2, \dots, diffPS$) is the population size after $p - 1$ reductions. If g_r is the generation whose population is to be reduced, then:

$$NP_{g+1} = \begin{cases} \frac{NP_g}{2}, & \text{if } g = g_r, \\ NP_g, & \text{otherwise.} \end{cases} \quad (14)$$

One individual from the first half of the current population is paired with the individual at the same index in the second half. Their fitness values are compared, and the better-performing individual is placed into the reduced population, which serves as the parent population for the next generation. On average, one population size reduction requires $\frac{NP}{2}$ comparisons of fitness values and $\frac{NP}{4}$ copying of individuals.

In (Brest and Sepesy Mauřec 2008) the experiment showed that $NP_{init} = 200$ with $diffPS = 4$ gave the best results for $D = 30$ and $NP_{init} = 400$ with $diffPS = 5$ for $D = 60$.

2.3.3 FDSADE (Tirronen and Neri 2009)

Tirronen and Neri (2009) proposed the Fitness Diversity Self-Adaptive Differential Evolution algorithm (FDSADE). Population size is adjusted according to parameter ϕ :

$$\phi = \frac{\sigma_f}{|f_{worst} - f_{best}|} \quad (15)$$

where σ_f is the standard deviation of the fitness values of individuals, f_{worst} and f_{best} are the worst and best fitness values, respectively. Population size is modified for the next generation according to the equation:

$$NP_{g+1} = NP_{min} + NP_v \times (1 - \phi) \quad (16)$$

where NP_{min} stands for fixed population size and is the minimum value of the population size, NP_v stands for variable population size and is the maximum value of the variable contribution of the population size. The new value of NP_{g+1} is then employed for the next generation: if population size is reduced, only those NP_{g+1} individuals having the best performance are considered for the next steps; if the population is expanded, the best solutions are duplicated in order to have a population composed of NP_{g+1} individuals. In the experimental setting, the authors used the following parameter values: $NP_{min} = 10$ and $NP_v = 40$.

2.3.4 ADE (Zhabitsky and Zhabitskaya 2013)

Rather than addressing the challenge of determining an appropriate population size, which depends on the complexity of the function being minimized, Asynchronous Differential Evolution (ADE) (Zhabitsky and Zhabitskaya 2013) with independent restart was employed. To balance fast convergence with a high likelihood of locating the global minimum, the algorithm begins with a small initial population size NP_{init} . When ADE detects stagnation in the search for better solutions, it triggers a restart by increasing the population size using a predefined multiplier k and initiating an independent search. Reasonable values of k typically lie in the interval $[1.2, 3.0]$. In their work, they set $k = 2$ and $NP_{init} = 10$.

2.3.5 L-SHADE (Tanabe and Fukunaga 2014)

Tanabe and Fukunaga (2014) introduced an improved version of the SHADE (Tanabe and Fukunaga 2013) algorithm, called L-SHADE, which uses linear population size reduction (LPSR) to dynamically reduce the population size during the evolution process. This represents a simple, deterministic method for population size reduction that operates with only one control parameter: NP_{init} , which represents the initial population size. After each generation g , the population size for the next generation NP_{g+1} is calculated as:

$$NP_{g+1} = \text{round} \left(\left(\frac{NP_{end} - NP_{init}}{MaxFEs} \right) \times nfe + NP_{init} \right), \quad (17)$$

where $MaxFEs$ denotes the maximum number of fitness function evaluations that the algorithm has available, nfe denotes the current number of fitness function evaluations, and NP_{end} denotes the final population size and is set to 4. When the population size is decreased, the worst individuals are removed from the population until the target population size is reached.

Using Eq. (17), the population is linearly decreased from NP_{init} to NP_{end} based on the current progress of the optimization process. At the beginning of the optimization process initial size of the population is set to $NP_{init} = \text{round}(18 \times D)$.

Apart from the introduction of dynamic population size adjustment, all other aspects of the SHADE algorithm remain unchanged. This includes mechanisms like the historical memory for control parameters, the archive of individuals, and the mutation strategy.

3 Review methodology

This study employs a scoping review approach to identify and analyze relevant research works. The inclusion and exclusion criteria used for study selection are presented below. The review process generally follows the PRISMA 2020 framework (Page et al. 2021). The selection process involved searching and screening relevant research across multiple bibliographic databases. Three major digital repositories were utilized: Scopus, IEEE Xplore, and Web of Science (WoS). The review process was structured into four phases: identification, screening, eligibility, and inclusion.

A search for the period 2017–2026 was conducted on May 25th, 2026 (i.e., we included a period of the first five months of 2026):

- **Scopus:** TITLE-ABS-KEY ("differential evolution") AND TITLE-ABS-KEY ("population size")
- **IEEE Xplore:** ("All Metadata":"differential evolution") AND ("All Metadata":"population size")
- **Web of Science:** TS=("differential evolution") AND TS=("population size")

The flow diagram is shown in Figure 1. Specifically, this review considered all published articles matching the search keywords from early 2017 to May 2026. The database search identified 1,452 records: 484 from Scopus, 506 from IEEE Xplore, and 462 from Web of Science (WoS). Additionally, 16 records were added, obtained through snowballing.

The next stage involved screening the titles, abstracts, and keywords of the retrieved records. The procedure was conducted as follows: first, the title, abstract, and keywords of each article were evaluated against the inclusion criteria. If eligibility could not be determined based on this information alone, the full text of the article was reviewed. Following the title and abstract screening, 764 records were excluded, reducing the number of potentially relevant papers to 251.

We were unable to retrieve 3 full-text reports. After a full-text assessment of the remaining 248 reports using the predefined inclusion and exclusion criteria, 44 reports were excluded, while 204 studies met the inclusion criteria and were included in the final set.

In our evaluation, we did not differentiate between studies published in journals and those published at conferences. Therefore, our final set contains journal and conference articles.

Inclusion criteria: A study was included if it:

- focused on Differential Evolution (DE),

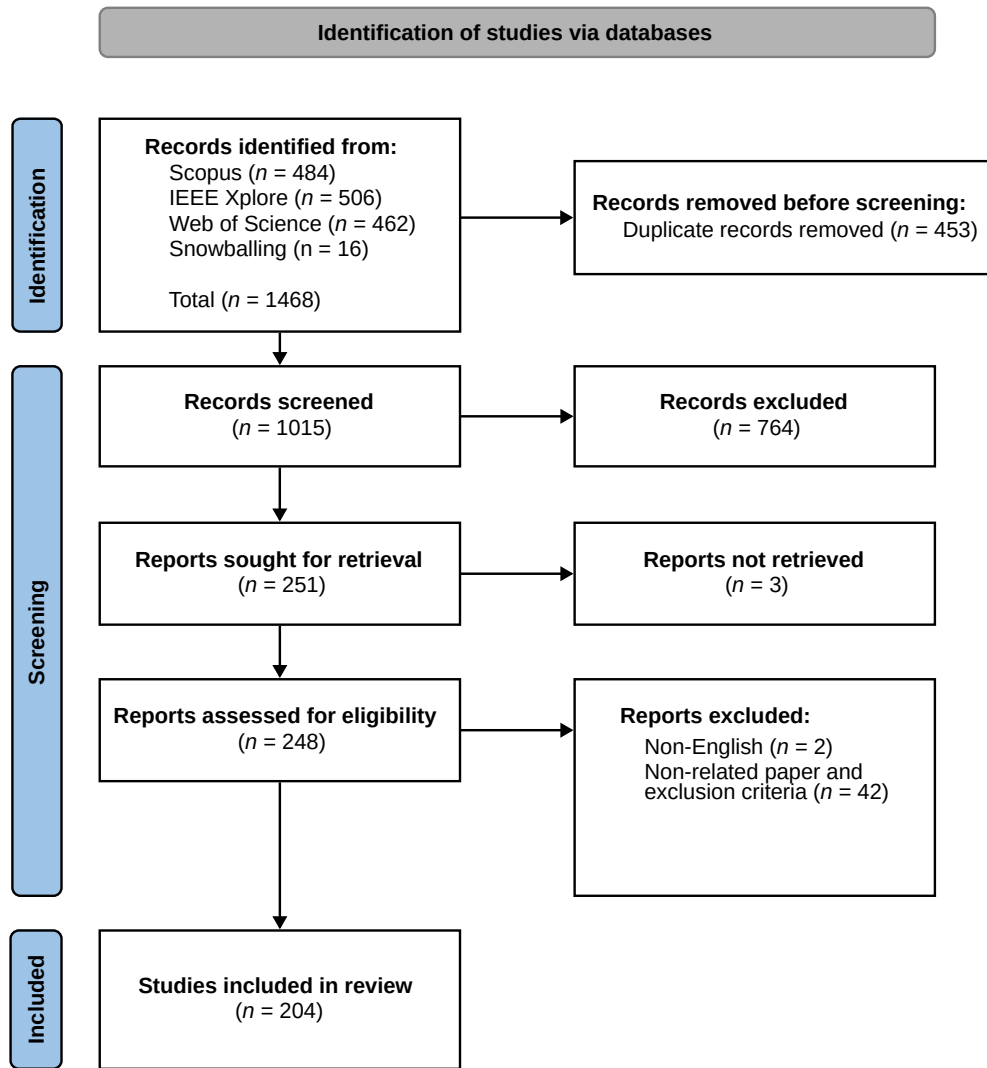


Fig. 1: PRISMA flow diagram for the selection of papers retrieved from databases.

- employed a population size adjustment mechanism during the evolutionary process,
- was published in a peer-reviewed journal, conference proceedings, or a book chapter from 2017 onwards.

Exclusion criteria: Studies were excluded if they:

- were not written in English,
- included Differential Evolution only in their experimental comparison,

- partitioned a large population into multiple subpopulations (either before or during the evolutionary process),
- used a dynamic population size to encode a variable number of points representing a single solution (e.g., in UAV path planning problems).

Two papers were excluded since they were not written in English. The final set includes 204 papers that met the inclusion criteria. During the analysis, the selected studies were classified into two groups according to the type of population size adjustment employed:

- linear, and
- non-linear.

A total of 148 papers were assigned to the first group and 56 papers to the second. The population size adjustment mechanisms proposed in these groups are discussed in detail in the following section.

4 Population size adjustment in DE-based algorithms

In this section, an overview of DE-based algorithms that utilize population size adjustment published from 2017 onwards is given. The adjustment of the population size influences both the algorithm’s search capability and computational cost. Mechanisms in which the population size decreases during the evolutionary process are more commonly used in the literature than mechanisms in which the population size increases. In some research works, both mechanisms are used together, i.e., an algorithm can decrease and increase the population size depending on how the optimization process is progressing.

The section is divided into two subsections. The algorithms published from 2017 onwards are grouped according to the primary mechanism used for population size adjustment:

- Linear population size adjustment in DE-based algorithms (Section 4.1),
- Non-linear population size adjustment in DE-based algorithms (Section 4.2).

It should be noted that this classification is not always straightforward. Some algorithms combine multiple population adjustment mechanisms, such as fixed population phases followed by linear or non-linear reduction. Others employ heuristic strategies that adapt the population size based on indicators such as search stagnation, convergence progress, or population diversity. Consequently, the proposed classification is based on the dominant population adjustment mechanism of each algorithm, although several methods incorporate features that span multiple categories. The descriptions presented in this section focus primarily on the population size adjustment mechanisms. Other components of the reviewed algorithms are omitted or discussed only briefly when necessary to provide context for the proposed population adaptation strategy.

4.1 Linear population size adjustment in DE-based algorithms

In this subsection, we present a small subset of DE-based algorithms incorporating a linear population size adjustment mechanism published since 2017, and for clarity, the reviewed algorithms are organized chronologically by year of publication. Most of these algorithms represent the current state-of-the-art and typically combine LPSR with additional mechanisms, such as adaptive control parameter tuning. However, numerous other algorithms and studies have employed these approaches to address a wide range of real-world optimization problems. A comprehensive overview of these studies is provided in Tables B23 in the Appendix, where the publication type is also indicated for each reference. The collected literature comprises 107 journal articles and 41 conference papers.

4.1.1 DMPSA mechanism (Poláková et al. 2019)

In (Poláková et al. 2019), population diversity is used to adapt the population size in DE. The population size is adjusted based on the current relative diversity RD (see Eqs. (18) and (19)) and the proportion of depleted function evaluations $RFES$ (see Eq. (20)). The authors suggest keeping the relative diversity close to the target value rRD , which decreases linearly from 1 ($RFES = 0$) at the beginning to 0 ($RFES = 0.9$) near the end of the search process. The population size is increased by 1 (adding a random solution) when $RD < 0.9 \times rRD$ and decreased by 1 (removing the worst solution) when $RD > 1.1 \times rRD$.

$$DI = \sqrt{\frac{1}{NP} \sum_{i=1}^{NP} \sum_{j=1}^D (x_{i,j} - \bar{x}_j)^2} \quad (18)$$

$$RD = \frac{DI}{DI_{init}} \quad (19)$$

$$RFES = \frac{n_{fes}}{MaxFEs} \quad (20)$$

The search process should start with an initial population size NP_{init} and, during optimization, remains within the range NP_{min} to NP_{max} . The authors recommend the following values: $NP_{init} = 50$, $NP_{min} = 8$, and $NP_{max} = 5 \times D$.

4.1.2 MjSO (Li et al. 2023b)

Algorithm MjSO (Li et al. 2023b) extends the jSO algorithm with three key modifications. Firstly, the selection mechanism of p -best individuals, used in the mutation, is modified so that it assigns a selection probability to each individual. Rather than treating all p -best individuals equally as is in the base jSO, MjSO assigns a higher selection probability to better individuals using a probability distribution controlled by a probability scale factor m and the individuals' rank.

Additionally, MjSO replaces the traditional mutation strategy DE/current-to- p Best-w/1 with a directed mutation strategy:

$$\mathbf{v}_i = \begin{cases} \mathbf{x}_i + F_w \times (\mathbf{x}_{pBest} - \mathbf{x}_i) + F \times (\mathbf{x}_{r_1} - \mathbf{x}_{r_2}), & f(\mathbf{x}_{r_1}) \leq f(\mathbf{x}_{r_2}) \\ \mathbf{x}_i + F_w \times (\mathbf{x}_{pBest} - \mathbf{x}_i) + F \times (\mathbf{x}_{r_2} - \mathbf{x}_{r_1}), & f(\mathbf{x}_{r_1}) > f(\mathbf{x}_{r_2}) \end{cases} \quad (21)$$

The external archive is also omitted to avoid excessive use of historically dominant individuals. Individuals $\mathbf{x}_{r_1}$ and $\mathbf{x}_{r_2}$ are therefore selected randomly from the current population.

During the evolution, the population size is reduced using the standard LPSR mechanism from $NP_{init} = 25 \times \log(D) \times \sqrt{D}$ to $NP_{end} = 4$.

4.1.3 APSM-jSO (Li et al. 2023c)

APSM-jSO (Li et al. 2023c) algorithm is an enhanced version of jSO algorithm. It introduces simple and effective modifications to the selection of the scaling factor, crossover rate, and external archive update mechanism. An external archive is utilized alongside linear population-size reduction. Within the jSO algorithm, the size of the external archive is constrained by a predefined capacity, defined as $|A| = NP \times A_{rate}$. Once this limit is reached, a randomly selected individual is removed to accommodate a newly added one. However, due to this random update mechanism, certain archive members may persist for many generations, leading to their excessive reuse—particularly in the later stages of optimization. To mitigate this issue and promote population diversity, APSM-jSO adopts a first-in, first-out (FIFO) strategy for archive management. Under this scheme, when the archive reaches its maximum size, the newly inserted individual replaces the one that has been stored the longest. In the experiment performed by the authors of the algorithm, the size of initial population was set to $NP_{init} = 75 \times D^{\frac{2}{3}}$.

4.1.4 cDE-APAR (Sun et al. 2023b)

Sun et al. (2023b) proposed a cascaded differential evolution optimization framework with adaptive population allocation and reduction, and the algorithm is called cDE-APAR. The algorithm improves the performance of a single algorithm by cascading two DE variants and combining a local search strategy and a global search strategy to further improve exploitation and exploration. Further, a local search behavior, and a global search behavior are executed in parallel in the second stage. To allocate individuals into sub-populations, an adaptive strategy is used. As reduction, the classic LPSR strategy is adopted to eliminate individuals with poor fitness.

4.1.5 L-SRTDE (Stanovov and Semenkin 2024c)

The Linear population size reduction Success RaTe-based Differential Evolution algorithm (L-SRTDE) (Stanovov and Semenkin 2024c) is the winning algorithm of the CEC 2024 competition on single-objective bound-constrained optimization (Qiao et al. 2023). A key feature of L-SRTDE is the utilization of the success rate SR of individuals for the adaptation of the scaling factor F and the value of the parameter p . The mean value used for sampling the scaling factor is calculated using a hyperbolic tangent function as:

$$mF = 0.4 + 0.25 \times \tanh(5 \times SR), \quad (22)$$

and the actual scaling factor F is then sampled with Normal distribution as:

$$F = \mathcal{N}(mF, 0.02). \quad (23)$$

Additionally, the success rate is used to set the value of p , which controls the selection pressure when choosing the $pBest$ individual:

$$p = 0.7 \times e^{-7 \times SR}. \quad (24)$$

To retain information about good individuals, L-SRTDE maintains two populations of the same size during the optimization process. The population $\mathbf{P}^{new}$ contains the individuals of the current generation, while $\mathbf{P}^{top}$ contains the best individuals obtained so far. Mutation strategy DE/r-new-to-ptop/n/t, used in L-SRTDE, differs slightly from standard DE/current-to-pBest-w/1:

$$\mathbf{v}_i = \mathbf{x}_{r_1}^{new} + F \times (\mathbf{x}_{pBest}^{top} - \mathbf{x}_i^{new}) + F \times (\mathbf{x}_{r_2}^{new} - \mathbf{x}_{r_3}^{top}), \quad (25)$$

here $\mathbf{x}^{new}$ represents an individual taken from $\mathbf{P}^{new}$, and $\mathbf{x}^{top}$ represents an individual taken from $\mathbf{P}^{top}$.

Both populations are subjected to linear population-size reduction (LPSR), decreasing from NP_{init} to NP_{end} . The authors recommend $NP_{init} = 20 \times D$ and $NP_{end} = 4$, the latter is the smallest population size with which the proposed mutation can be carried out.

4.1.6 DSDCDE (Deng et al. 2025)

Population reduction strategies typically use only one evaluation criterion—based solely on fitness—to decide which individuals to remove. As a result, individuals tend to accumulate around local optima, while potentially valuable candidates may be discarded too early. This weakens the algorithm’s exploratory and exploitative abilities and increases the likelihood of becoming trapped in local optima. The dual-space-based population size adaptation (DSPSA) (Deng et al. 2025) method exploits both fitness- and search-space information to adapt the population size. It adaptively adjusts population size using dual-space information. This makes it possible to eliminate individuals with limited potential while preserving those that are more promising, which in turn enhances the algorithm’s capability to avoid local optima. Its first implementation appeared in the DSDCDE algorithm (Deng et al. 2025).

The DSPSA begins by computing the cosine similarity between each individual in the current population $\mathbf{x}_{i,g}$ and the current best individual $\mathbf{x}_{best,g}$:

$$CS_i = CS(\mathbf{x}_{i,g}, \mathbf{x}_{best,g}) = \frac{\mathbf{x}_{i,g} \times \mathbf{x}_{best,g}}{\|\mathbf{x}_{i,g}\| \times \|\mathbf{x}_{best,g}\|} = \frac{\sum_{j=1}^D x_{i,g}^j \times x_{best,g}^j}{\sqrt{\sum_{j=1}^D (x_{i,g}^j)^2} \times \sqrt{\sum_{j=1}^D (x_{best,g}^j)^2}}. \quad (26)$$

Next, the fitness values and cosine similarity values are sorted—fitness in ascending order and cosine similarity in descending order—yielding two separate rankings for each individual: a fitness ranking $Rank_i^{fitness}$ and a cosine-similarity ranking $Rank_i^{cs}$. These two ranks are then combined to produce a final rank for each individual:

$$Rank_i = Rank_i^{fitness} + Rank_i^{cs}. \quad (27)$$

Individuals with the lowest ranks are deleted. The population size for the next generation is determined by:

$$NP_{g+1} = \left\lfloor NP_{init} + \frac{(NP_{end} - NP_{init})}{MaxFEs} \times nfes \right\rfloor. \quad (28)$$

NP_{init} is set to $18 \times D$, and NP_{end} is set to 4. The population is divided according to the individuals' fitness values into three groups: an elite subpopulation ($\mathbf{P}_e$), a mediocre subpopulation ($\mathbf{P}_m$), and an inferior subpopulation ($\mathbf{P}_i$). The relationships between the full population and these subpopulations are given by the following equations:

$$\mathbf{P} = \mathbf{P}_e \cup \mathbf{P}_m \cup \mathbf{P}_i. \quad (29)$$

The individuals are first sorted according to their fitness values, after which the population is dynamically partitioned into three subpopulations. Depending on the chosen proportions, the population is divided into an elite group, a mediocre group, and an inferior group, with sizes $100p\% \times NP$, $(1 - 2 \times 100p\%) \times NP$, and $100p\% \times NP$, respectively. Consequently, the relationship between the overall population size and the sizes of the three subpopulations can be expressed as follows:

$$NP = NP^{EP} + NP^{MP} + NP^{IP}. \quad (30)$$

NP^{EP} is the size of the elite group, NP^{MP} the size of mediocre group and NP^{IP} the size of the inferior group. The parameter p varies adaptively throughout the evolution, and its value is determined as follows:

$$p = p_{min} + \frac{p_{max} - p_{min}}{MaxFEs} \times nfes, \quad (31)$$

where parameter $p_{max} = 0.25$ and $p_{min} = 0.125$ are set as in jSO (Brest et al. 2017).

4.1.7 FODE (Wang et al. 2025a)

Wang et al. (2025a) proposed Fractional Order DE (FODE) to enhance DE performance from two aspects. First, it uses the bi-strategy co-deployment framework, where the population-based and parameter-based strategies are combined. Second, the fractional order (FO) calculus is applied to the differential vector to enhance DE's exploration ability by using the historical information of individuals. The FODE algorithm applies the LPSR strategy (see Eq. (10)) and uses the initial population size of $NP_{init} = 18 \times D$.

4.1.8 mLSHADE-SPACMA (Fu et al. 2025)

Modified LSHADE-SPACMA, called mLSHADE-SPACMA (Fu et al. 2025), is an upgrade of the CEC 2017 winning algorithm with a new mutation strategy and external archiving mechanism.

The mLSHADE-SPACMA algorithm introduces a more accurate mechanism for eliminating and generating individuals to enhance the algorithm's local exploitation ability. It also improves the mutation strategy based on a modified semi-parametric adaptive strategy and rank-based selective pressure, which improves the algorithm's evolutionary direction. Additionally, it uses the elite-based external archive mechanism.

LPSR is used in the mLSHADE-SPACMA algorithm with $NP_{init} = 18 \times D$ and $NP_{end} = 4$.

4.1.9 TPADE (Gong et al. 2025)

To address the problem of premature convergence, Gong et al. (2025) introduced Triple-population-based Adaptive Differential Evolution (TPADE), which improves evolutionary efficiency by dividing the current population based on objective values into three sub-populations: *superior*, *medium*, and *inferior*. It adopts the linear population size reduction (LPSR) mechanism from L-SHADE, while the sizes of the superior (SN_g), inferior (IN_g), and medium (MN_g) sub-populations are determined by the following equations:

$$IN_g = SN_g = \left\lfloor NP_g \times \left(p_{min} + (p_{max} - p_{min}) \times \frac{n_{fes}}{MaxFES} \right) \right\rfloor, \quad (32)$$

$$MN_g = NP_g - IN_g - SN_g, \quad (33)$$

where p_{min} and p_{max} are control parameters and NP_g is the population size in the g -th generation. Each sub-population uses distinct operators to maintain diversity and guide search directions, and superior trials can replace inferior individuals, adding valuable directional information for the next generation.

As already mentioned, TPADE uses the LPSR mechanism with $NP_{init} = 18 \times D$ and $NP_{min} = 10$.

4.1.10 L-SRTDEwPSM (Popič et al. 2025)

Algorithm L-SRTDEwPSM (Popič et al. 2025) further extends L-SRTDE with the incorporation of a p -best selection mechanism (PSM). This mechanism provides a stricter criterion for selecting the p -best individual to be used in the mutation strategy by enforcing a minimum distance r between the individual selected as p -best and all other better individuals. This ensured that candidates suitable for $\mathbf{x}_{pBest}$ are not clustered together in the search space, thereby increasing the algorithm's exploration potential. In L-SRTDEwPSM the minimal distance r is calculated based on

the problem dimension D and control parameter R as:

$$r = \frac{\sqrt{D}}{R}, \quad (34)$$

where authors have set $R = 5000$.

As in the base L-SRTDE, this algorithm still uses two separate populations that are linearly reduced with LPSR from $NP_{init} = 20 \times D$ to $NP_{end} = 4$.

4.2 Non-linear population size adjustment in DE-based algorithms

This section provides an overview of DE-based algorithms that incorporate the NLPSR mechanism for population size adjustment. A total of 56 algorithms published since 2017 are reviewed. The corresponding publications are summarized in Table B24 in the Appendix, where the publication type is indicated for each reference. The surveyed literature comprises 45 journal articles, 10 conference papers, and 1 book chapter. For clarity, the reviewed algorithms are organized chronologically by year of publication.

4.2.1 CAPR (Wong et al. 2017)

Continuous Adaptive Population Reduction (CAPR), proposed by Wong et al. (2017), gradually reduces the population size according to the change of gradient of the fitness value.

$$NP_{g+1} = \begin{cases} \sqrt[\alpha]{\frac{\Delta_g}{\Delta_{g-1}}}, & \text{if } 0 < \frac{\Delta_g}{\Delta_{g-1}} < 1, \\ NP_g, & \text{otherwise,} \end{cases} \quad (35)$$

$$\Delta_g = \frac{f_{avg}(\mathbf{x}_g) - f_{avg}(\mathbf{x}_{g-1})}{f_{avg}(\mathbf{x}_g)}, \quad \Delta_{g-1} = \frac{f_{avg}(\mathbf{x}_{g-1}) - f_{avg}(\mathbf{x}_{g-2})}{f_{avg}(\mathbf{x}_{g-1})}. \quad (36)$$

$f_{avg}(\mathbf{x}_g)$ denotes the average fitness function value of the individuals in the population at generation g , and α controls the population reduction rate. The authors set α to 100.

4.2.2 IDEbd (Bujok and Tvrdik 2017)

The population size in (Bujok and Tvrdik 2017) is adjusted dynamically based on the relative diversity of the population compared to a target linear decrease. Population diversity is computed as the root mean square deviation of individual from the population center:

$$Div = \sqrt{\frac{1}{NP} \sum_{i=1}^{NP} \sum_{j=1}^D (x_{i,j} - \bar{x}_j)^2}, \quad (37)$$

where $\bar{x}_j$ is the mean of the j -th dimension. Ideally, relative diversity, defined as current diversity divided by initial diversity, should decrease linearly with the relative

number of function evaluations. The authors allowed a tolerance band of ± 0.1 around this linear trend. If the current diversity falls below the expected level, a new random individual is added to increase diversity, and if diversity exceeds the expected level, the worst individual is removed.

4.2.3 EsDEr-NR (Awad et al. 2018)

The algorithm Ensemble sinusoidal DE with restart strategy and niching-based population size reduction (EsDEr-NR) (Awad et al. 2018) introduces three key modifications to L-SHADE algorithm. Firstly, it employs an ensemble sinusoidal scheme for adapting the scaling factor F during the first half of the maximum number of generations. For the second half of the generation, the Cauchy distribution is used to adapt the scaling factor F .

Secondly, when the population size is reduced to 20 individuals, they perform their restart strategy, in which the 10 worst individuals are reinitialized. Then the Gaussian walk is conducted to generate 10 new individuals from the reinitialized ones.

Lastly, EsDEr-NR utilizes a niching-based population size reduction. Population size remains unchanged for the first 50% of $MaxFEs$, then the Euclidean distance between the best individual $\mathbf{x}_{best}$ and every other individual is calculated, and the population is sorted based on this distance. The first half of the sorted population is grouped in the first niche, and the second half of the sorted population is grouped in the second niche; here, both niches contain $\frac{NP}{2}$ individuals. Then, the new target population size is calculated according to Eq. (38), and individuals closest to the best individual from the second niche are removed.

$$NP_{g+1} = \text{round} \left(\frac{NP_{min} - NP_{init}}{MaxFEs} \times nfes + NP_{init} \right) \quad (38)$$

In their paper, the authors used $NP_{init} = 18 \times D$ and $NP_{min} = 4$.

4.2.4 dyn-NP in LSGO (Maucec et al. 2018)

In (Maucec et al. 2018) the dyn-NP mechanism (see Section 2.3.2) was applied to solve the large-scale global optimization (LSGO) problems. Parameters in the experiments were set as follows: $NP_{init} = 3200$ and $NP_{end} = 50$.

4.2.5 EAGDE (Mohamed and Mohamed 2019)

Mohamed and Mohamed (2019) proposed the Enhanced Adaptive Guided Differential Evolution algorithm (EAGDE). It employs a non-linear population size reduction, which gradually decreases the population size according to a non-linear function:

$$NP_{g+1} = \text{round} \left[NP_{init} + (NP_{end} - NP_{init}) \times \left(\frac{nfes}{MaxFEs} \right)^X \right], \quad (39)$$

where X is calculated as:

$$X = 1 - \left(\frac{nfe}{MaxFEs} \right). \quad (40)$$

The main idea of this population reduction technique is to delete a larger number of bad solutions during the early-to-middle stages of the evolutionary process without affecting the algorithm's exploration capability, and to delete a smaller number of individuals during the middle-to-late stages to enhance the algorithm's exploitation tendency (Mohamed and Mohamed 2019).

The EAGDE algorithm was tested by solving the CEC 2013 benchmark problems. The initial population size NP_{init} was set to $10 \times D$, $6 \times D$ and $4 \times D$ for the dimensions 10, 30 and 50, respectively.

4.2.6 HARD-DE (Meng and Pan 2019)

The algorithm Hierarchical ARchive Based Mutation Strategy With Depth Information of Evolution (HARD-DE) (Meng and Pan 2019) employs a parabolic population size reduction scheme to slow down the decrease of population size at the beginning of the evolution. In general, the parabola is required to pass through two points: (NP_{init}, NP_{init}) and $(MaxFEs, NP_{end})$.

Using the $Pivot = (x, y)$, where x -axis denotes the current number of function evaluations and y -axis denotes the current population size, the population size reduction scheme is defined as:

$$NP_{g+1} = \begin{cases} \left\lfloor \frac{y - NP_{init}}{(x - NP_{init})^2} \times (nfe - NP_{init})^2 + NP_{init} \right\rfloor, & \text{if } nfe \leq x, \\ \left\lfloor \frac{y - NP_{init}}{(x - NP_{end})} \times (nfe - MaxFEs) + NP_{end} \right\rfloor, & \text{otherwise,} \end{cases} \quad (41)$$

where

$$x = \frac{2}{3} \times MaxFEs, \quad y = \frac{1}{3} \times NP_{init}.$$

It is a combined parabolic-linear population size reduction scheme. When $Pivot = (MaxFEs, NP_{end})$, the pivot-based population size reduction strategy simplifies to a parabolic reduction scheme. Conversely, when $Pivot = (NP_{init}, NP_{init})$, the pivot-based approach reduces to a linear population size reduction scheme. The initial population size NP_{init} and the final population size NP_{end} are set following the same procedure as in jSO.

4.2.7 PaDE (Meng et al. 2019a)

A rapid decrease in population size at the beginning of evolution—as used in linear population-reduction schemes—can impair the algorithm's ability to explore the search space and to perceive the landscape of certain objective functions. To address this problem, the PaDE (Meng et al. 2019a) algorithm uses a parabolic population-size reduction model similar to that described in Section 4.2.6. This parabolic reduction model is shown in Eq. (42).

$$NP_{g+1} = \begin{cases} \left\lceil \frac{y - NP_{init}}{(x - NP_{init})^2} \times (nfes - NP_{init})^2 + NP_{init} \right\rceil, & \text{if } nfes \leq x, \\ \left\lceil \frac{y - NP_{init}}{x - NP_{end}} \times (nfes - MaxFEs) + NP_{end} \right\rceil, & \text{otherwise.} \end{cases} \quad (42)$$

Here, the $Pivot = (x, y) = (\frac{2}{3} \times MaxFEs, \frac{1}{3} \times NP_{init})$ represents the default setting for the adaptation of population size. The parameters recommended by the authors are $NP_{init} = 25 \times \log(D) \times \sqrt{D}$ and $NP_{end} = 4$.

4.2.8 ESHADE_USM (Kumar et al. 2019)

Kumar et al. (2019) proposed an enhanced version of the multi-operators variant of DE (MODE). MODE employs multiple mutation and crossover strategies within a single DE algorithm structure. In the enhanced version of MODE, named ESHADE_USM, in addition an exponential population size reduction (EPSR) technique is used:

$$NP_{g+1} = NP_{init} \times \text{round} \left(1 - \gamma \times \frac{NP_{init} - NP_{end}}{MaxFEs} \right)^g. \quad (43)$$

γ is a parameter to control the exponential curve. NP_{end} equals to 4 and NP_{init} equals to $18 \times D$.

4.2.9 PE-SHADE'19 (Zhang et al. 2019)

An enhanced algorithm with a population entropy-based population adaptation strategy has been proposed under the framework of SHADE (PE-SHADE) (Zhang et al. 2019). A method to calculate the entropy of the population is introduced, and the population size is adapted according to the distribution state with a population increasing strategy and a population reduction strategy.

For each generation g , the population entropy value E_g is calculated. For a description of how the population entropy is calculated, see (Zhang et al. 2019). Here, the population entropy value is defined by the multiplication rather than the more common summation.

The ratio R is expressed as follows:

$$R = \frac{E_{g+1}}{E_g}. \quad (44)$$

If $R > 1$, the population entropy is increasing, which means that the population has a tendency to diverge within the search space. On the contrary, if $R < 1$, the individuals are converging in the search space.

The population size is increased until $0.2 \times MaxFEs$ function evaluations are used; thereafter, the reduction strategy is applied. The population size varies in a given range, and the upper bound NP_{max} and lower bound NP_{min} of the population size are

initialized as $10 \times D$ and D , respectively. The NP_{init} was set to be 100. The average value of the R , denoted by R_{avg} , is calculated for every N generations (using the modulo operation $g \bmod N = 0$).

When $nfes < 0.2 \times MaxFEs$ then $rate_1 \times NP_g$ individuals are randomly generated if $rand(0, 1) > R_{avg}$. If the population size exceeds the upper bound NP_{max} , it is truncated to NP_{max} . When the number of evaluations $nfes \geq 0.2 \times MaxFEs$ then the worst $rate_2 \times NP_g$ individuals are removed from the population if $rand(0, 1) < R_{avg}$. In order to avoid the population size exceeding the lower bound, the population size can be reduced at most to the lower bound. The parameter values for $rate_1$, $rate_2$, and N were 0.3, 0.05, and 4, respectively.

4.2.10 eL-SHADE (Khan et al. 2020)

Khan et al. (2020) proposed Enhanced Linear population size reduction technique of Success History based Adaptive Differential Evolution (eL-SHADE). The eL-SHADE algorithm employs the NLPSR scheme, i.e., a parabolic population size reduction (PPSR), as defined in Eq. (45).

$$NP_{g+1} = \left\lceil \frac{NP_{end} - NP_g}{(G - NP_g)^2} \times (g - NP_g)^2 + NP_g \right\rceil \quad (45)$$

In their experiments, the initial population size was set to $NP_{init} = 500$, the final size to $NP_{end} = 150$, and the maximum number of generations to $G = 5000$.

4.2.11 SAEDE-EP (Budiman et al. 2020)

The SAEDE-EP algorithm (Budiman et al. 2020) employs an adaptive population sizing mechanism based on the stagnation of the best solution. The stagnation level is measured by the variable $q_{best,g}$, which counts the number of consecutive generations without improvement of the best individual

$$q_{best,g} = \begin{cases} 0, & \text{if } f(\mathbf{x}_{best,g}) < f(\mathbf{x}_{best,g-1}), \\ q_{best,g-1} + 1, & \text{otherwise.} \end{cases} \quad (46)$$

When the stagnation counter exceeds a predefined threshold, the population size adaptation mechanism is activated. A growth factor R_{g+1} is selected either from a predefined pool of candidate values or from a memory of previously successful values. Reader interested in details is referred to original paper (Budiman et al. 2020). The population size for the next generation is then calculated as:

$$NP_{g+1} = \text{int}(R_{g+1} \times NP_g + NP_g + 0.5). \quad (47)$$

If no significant stagnation is detected, the population size remains unchanged. The main objective of this mechanism is to increase population diversity when the search process stagnates, while preserving efficient population sizes when the optimization progresses successfully.

4.2.12 APDE (Zhang et al. 2020)

Particularly in large-scale global optimization (LSGO) problems, it is important to efficiently utilize all generated solutions and maintain sufficient population diversity under a limited fitness evaluation budget. APDE, proposed by Zhang et al. (2020), employs a dual population control strategy consisting of population increase (*pop_inc*) and population decrease (*pop_dec*).

When the best solution remains unchanged for a predefined number of iterations, inferior trial vectors are allowed to enter the population, increasing diversity and helping the search escape stagnation. If the $\mathbf{x}_{best}$ has not been changed for T iterations and NP is smaller than NP_{max} , the trial vector $\mathbf{u}_i$ is added to the population. In (Zhang et al. 2020), T was set to 15.

To prevent excessive population growth and maintain computational efficiency, a population reduction strategy periodically removes ineffective individuals based on their fitness and stagnation history. To decide which individual to be deleted, a factor named "degradation value" (dg) of individual is defined as:

$$dg(\mathbf{x}_i) = \frac{f(\mathbf{x}_i) - f(\mathbf{x}_{best}) + 1.0}{f(\mathbf{x}_{worst}) - f(\mathbf{x}_{best}) + 1.0} \times notChange_i. \quad (48)$$

$notChange_i$ records how many iterations the individual $\mathbf{x}_i$ has not changed. A high degradation value $dg(\mathbf{x}_i)$ suggests that the fitness of $\mathbf{x}_i$ is close to that of the worst individual or that the solution has remained unchanged for many generations. Consequently, $\mathbf{x}_i$ is regarded as ineffective and is removed from the population. If NP reduces to NP_{min} , the deletion process stops.

The *pop_inc* and *pop_dec* strategies ensure that the population size is maintained between NP_{min} and NP_{max} . In their experiments, NP_{min} was set to 50 and NP_{max} to 100.

4.2.13 PaDE-NPC (Meng et al. 2020b)

Meng et al. (2020b) proposed the PaDE-NPC algorithm. The population at the early stage of the evolution process is kept fixed, then, when $nfes > plat \times MaxFEs$, LPSR is applied. If we combine these two curves, a new non-linear curve is generated. The detailed equation of population size reduction in PaDE-NPC is:

$$NP_{g+1} = \left\lfloor \frac{nfes - plat \times MaxFEs}{(1 - plat) \times MaxFEs} \times (NP_{end} - NP_{init}) + NP_{init} \right\rfloor. \quad (49)$$

In their experiments $plat$ was set to 0.15, $NP_{init} = 25 \log(D) \sqrt{D}$ and $NP_{end} = 4$.

4.2.14 DE-NPC (Meng et al. 2020a)

DE-NPC, proposed by Meng et al. (2020a), uses a combined parabolic-linear population size reduction mechanism with $pivot = (x, y)$, as proposed in HARD-DE (see Section 4.2.6). When $pivot = (MaxFEs, NP_{end})$, the population size reduction is degraded into the parabolic population size reduction scheme and when

$pivot = (NP_{init}, NP_{init})$, the population size reduction is degraded into a linear population size reduction scheme. Furthermore, the population update and external archive maintenance are performed in the same manner as in L-SHADE. In their experiments the authors used $NP_{init} = 25 \times \log(D)\sqrt{D}$ and $NP_{end} = 4$.

4.2.15 ADDE (Zhan et al. 2020)

An adaptive distributed differential evolution (ADDE), proposed by Zhan et al. (2020), uses three populations named exploration population, exploitation population, and balance population. They are co-evolved concurrently by using the master-slave multipopulation distributed framework.

In ADDE, population size is adjusted based on the best solution improvement:

$$\frac{|f(\mathbf{x}_{best,g}) - f(\mathbf{x}_{best,g-CG})|}{|f(\mathbf{x}_{best,g-CG})|}. \quad (50)$$

$f(\mathbf{x}_{best,g})$ is the fitness of the current best individual, $f(\mathbf{x}_{best,g-CG})$ is the fitness of the best individual CG generations earlier. The authors set $CG = 30$ generations. When the improvement of the best solution exceeds the threshold θ , the population size is decreased by removing the NG worst individuals and storing them in the archive, reducing the population size to $NP - NG$. If the improvement of the best solution is below the threshold θ , the algorithm is considered to be stagnating. In this case, NG individuals are randomly selected from the archive and added to the population, increasing its size from NP to $NP + NG$ and enhancing population diversity. Authors set $NG = 0.4 \times D$. As the evolutionary process progresses, the improvement of the best solution typically becomes smaller. Therefore, the threshold θ is gradually reduced during the search using the exponential model:

$$\theta = 10^{-1-4 \times n_{fes} / MaxFEs}. \quad (51)$$

NP_{min} and NP_{max} are used to control the bound of the population size. In their experiments, $NP_{max} = 10 \times D$ and $NP_{min} = 2 \times D$.

4.2.16 FADE (Xia et al. 2021)

Xia et al. (2021) proposed a Fitness-based Adaptive Differential Evolution algorithm (FADE). In FADE, the entire population is split into multiple ($ss = 3$) small-sized populations and three popular breeding strategies are saved in an archive which can be utilized by the multiple populations. The population size can be adaptively adjusted during the evolutionary process according to the performance of the best individual.

During the evolutionary process, the population size needs to be within an interval $[NP_{min}, NP_{max}]$. The adaptation of population size in FADE includes two processes:

- deleting-individual process, and
- adding-individual process.

They are based on the successive generations of improvement of the global best individual and on the successive stagnation generations of the global best individual,

respectively. When deleting individuals from the current population, those individuals with the worst fitness are deleted from the population. The elites of all subpopulations in each generation are saved in an archive and used to generate a few promising individuals (Xia et al. 2021). Parameters in FADE are:

- $ss = 3$, $NP_{init} = 25 \times ss = 75$ for 30D problems, and $NP_{init} = 40 \times ss = 120$ for 50D problems.
- $MaxG_{imp} = MaxG_{stag} = 2$, which means that individuals with the worst fitness are deleted when the successive improvement generations of the best individual is larger than 2, and new individuals are added into the current population when successive stagnation generations of the best individual is greater than 2. In FADE, $nd = 6$ individuals are deleted from the population when performing the deleting-individual process, and $na = 3$ individuals are added into the population when conducting the adding-individual process. Note that 6 and 3 are divisible by subpopulation size ss .
- $NP_{min} = NP_{init}/3$ and $NP_{max} = NP_{init}$.

In FADE, after adding a few individuals to the current population, individuals in the population are randomly divided into multiple equal-sized subpopulations.

4.2.17 NL-SHADE-RSP (Stanovov et al. 2021)

The NL-SHADE-RSP (Stanovov et al. 2021) integrates several key mechanisms, including a rank-based selective pressure applied within the mutation strategy, adaptive utilization of an external archive, a dedicated set of control rules for dynamically regulating the crossover rate, and a non-linear population size reduction scheme defined as:

$$NP_{g+1} = \text{round} \left((NP_{end} - NP_{init}) \times \left(\frac{nfes}{MaxFES} \right)^{\left(1 - \frac{nfes}{MaxFES} \right)} + NP_{init} \right). \quad (52)$$

The NLPSR scheme employed in this work is similar to that proposed in EAGDE (Mohamed and Mohamed 2019). In their experiments, the initial population size was set to $NP_{init} = 30 \times D$ and the final size to $NP_{end} = 4$.

4.2.18 CS-DE (Meng et al. 2021)

The Cooperative Strategy based DE (CS-DE) (Meng et al. 2021) algorithm introduces an elliptic-linear population size reduction scheme. A fast reduction of the population size in the early phases of evolution can hinder exploration of the search space and, as a result, degrade optimization performance. To improve exploratory behavior, a more gradual decrease in population size is desirable. For this purpose, the CS-DE algorithm introduces a new population-size adaptation mechanism that combines elliptic and linear reduction strategies.

$$NP_{g+1} = \begin{cases} \left\lceil \sqrt{NP_{init}^2 - \frac{NP_{init}^2 - y^2}{x^2} \times nfes^2} \right\rceil, & \text{if } nfes < x, \\ \left\lceil \frac{NP_{init} - y}{MaxFEs - x} \times (nfes - x) + y \right\rceil, & \text{otherwise.} \end{cases} \quad (53)$$

$Pivot = (x, y)$ is the pivot that connects the ellipse and the line. Elliptic population-size reduction is applied during the early stage of evolution, whereas linear reduction is used in the later stage. CS-DE algorithm used the following values for x and y :

$$x = \frac{2}{3} \times MaxFEs, \quad y = \frac{1}{3} \times NP_{init}.$$

In their experiments, the initial population size was set to $NP_{init} = 25 \times \log(D) \times \sqrt{D} - 6$.

4.2.19 Crowding-based dynNP-DE (Zhong et al. 2021)

Zhong et al. (2021) proposed an improved crowding-based dynamic population size differential evolution (crowding-based dynNP-DE). It applies non-linear population size reduction dyn-NP-DE (see Section 2.3.2) proposed in (Brest and Sepesy Mauřec 2008).

4.2.20 Hip-DE (Meng and Yang 2021)

Hip-DE, defined in (Meng and Yang 2021), employs a population size adjustment mechanism similar to that of TDE; however, instead of using linear population size reduction (LPSR), it adopts a stepwise reduction scheme based on the number of function evaluations.

4.2.21 SLDE (Zeng et al. 2022)

Sawtooth-Linear Population Size Adaptive method (SLPSA) (Zeng et al. 2022) combines the characteristics of linear and sawtooth functions to improve the ability of the algorithm to jump out of a local optimum. It regulates population size using a sawtooth-linear control function. Although the overall population size decreases as the algorithm progresses, periodic injections of additional individuals cause temporary increases in population size. These individuals are drawn from an external archive that stores previously discarded trial vectors. Through this mechanism, SLPSA adaptively manages population diversity, enhancing both the exploration and exploitation capabilities of the algorithm. The population size update formula has the form:

$$NP_{g+1} = \text{round}(\alpha \times f_l + (1 - \alpha) \times f_l \times f_s), \quad (54)$$

$$f_l = NP_{init} - nfes \times \frac{(NP_{init} - NP_{end})}{MaxFEs}, \quad (55)$$

$$f_s = \text{sawtooth}(W \times g). \quad (56)$$

where, f_s denotes a bilaterally symmetric sawtooth function with a range of $[-1, 1]$, and W represents its period. The variable g corresponds to the current generation, f_l is a linear function, and α is a weighting coefficient constrained to the interval $[0, 1]$. In the original study, α was set to 0.95. When α takes a large value, the influence of the sawtooth function on population size becomes negligible. In the limiting case where $\alpha = 1$, SLPSA reduces to LPSR. Conversely, smaller values of α amplify the effect of the sawtooth function, leading to pronounced fluctuations in population size. SLPSA method was implemented in the SLDE algorithm. In their experiments, the initial population size was set to $NP_{init} = 14 \times D + 14$ and the final size to $NP_{end} = 4$.

4.2.22 DPSHADE (Li et al. 2022b)

Dynamic population reduction differential evolution algorithm (DPSHADE) is proposed in work (Li et al. 2022b). The dynamic population size reduction (DPSR) mechanism combines the linear and nonlinear population size reduction. It uses the nonlinear reduction to dynamically reduce the population size before half of the target function evaluations are used:

$$NP_{g+1} = \text{round}\left((NP_{end} - NP_{init}) \times M + NP_{init}\right), \quad (57)$$

where M is calculated as follows:

$$M = \frac{1}{1 + \exp\left(-\frac{10 \times n_{fes}}{MaxFEs} + 5\right)}. \quad (58)$$

After 50% of target function evaluations, the linear method is used:

$$NP_{g+1} = \text{round}\left(\frac{NP_{end} - NP_{init}}{MaxFEs} \times n_{fes} + NP_{init}\right). \quad (59)$$

In their experiments, the initial population size was set to $NP_{init} = 18 \times D$ and the final size to $NP_{end} = 4$.

4.2.23 ExR, LoR and PaR mechanisms (Najjarro et al. 2022)

For solving the localization problem of target nodes in wireless sensor networks, Najjarro et al. (2022) proposed three new population size adaptation mechanisms: exponential, parabolic, and logistic reduction.

In the exponential reduction (ExR) mechanism, the reduction of the population size follows an exponential decay as the generation index g increases:

$$NP_{g+1} = \text{round}\left((NP_{init} - NP_{end}) \times \beta^g + NP_{end}\right), \quad (60)$$

where $0 \leq \beta \leq 1$ is the control parameter of the ExR mechanism controlling how fast the population size reduction happens. The parameter β was set to 0.9.

The logistic reduction (LoR) mechanism has a smooth transition from exploration to exploitation by utilizing the population size reduction. It is conducted according to an inverse logistic function:

$$NP_{g+1} = \text{round}\left((NP_{init} - NP_{end}) \times \left(1 - \frac{1}{1 + e^{\left(\frac{G}{2} - g\right)\lambda}}\right) + NP_{end}\right), \quad (61)$$

where $0.1 \leq \lambda$ denotes the control parameter of the LoR mechanism controlling how smoothly the population size changes from the initial value NP_{init} to the lower bound NP_{end} . In their experiments parameter λ was set to 0.2.

In the parabolic reduction (PaR) mechanism, the population size reduction follows a parabolic decay during the evolution process:

$$NP_{g+1} = \text{round}\left(\left(\frac{NP_{init} - NP_{end}}{G^2}\right) \times (g - G)^2 + NP_{end}\right). \quad (62)$$

It is worth mentioning that, unlike the other two mechanisms described above, the PaR mechanism does not require control parameters.

4.2.24 NL-SHADE-RSP-MID (Biedrzycki et al. 2022)

Biedrzycki et al. (2022) proposed an enhanced version of NL-SHADE-RSP (Stanovov et al. 2021), named NL-SHADE-RSP-MID. In NL-SHADE-RSP the population size is reduced in the function of the utilized budget. To accelerate the reduction of the population size, NL-SHADE-RSP-MID replaces the original population update function with a homographic function:

$$NP_{g+1} = \left\lfloor a_1 + \frac{1}{b \times 10^{-6} + 2.57 \times a_2} \right\rfloor. \quad (63)$$

b is the number of evaluations performed during the current restart. a_1 and a_2 are the parameters set so that the population size equals 400 at the beginning of the restart ($b = 0$) and decreases to 20 at the end of the available evaluation budget for that restart.

4.2.25 IPM-DE (Morales-Castaneda et al. 2022)

Morales-Castaneda et al. (2022) employed Improved Population Management (IPM) to dynamically adjust the population size during the evolutionary process. IPM consists of two operators: the Stagnation Correction Operator (SCO) and the Exploitation Efficiency Operator (EEO).

The SCO operator is activated when the population diversity (Eq. 64) does not change significantly for a predefined number of generations, denoted by *LimitCount*. In this case, NP_{init} randomly initialized individuals are added to the population to increase diversity.

$$Div = \frac{1}{D} \sum_{j=1}^D Div_j \quad (64)$$

$$Div_j = \frac{1}{NP} \sum_{i=1}^{NP} |median(x_j) - x_{i,j}| \quad (65)$$

$median(x_j)$ is the median values of the j th dimension. The EEO operator compares the current population diversity (Eq. (64)) with the maximum diversity observed so far. Two scenarios are distinguished. When the current diversity falls below 10% of the maximum diversity, the algorithm is considered to be entering the exploitation phase, and the population size is reduced to NP_{begin} individuals. When the diversity falls below 2% of the maximum diversity, the algorithm is considered to be in a late stage of exploitation, and the population size is further reduced to NP_{late} individuals. In their experiments, the authors used $LimitCount = 10$, $NP_{begin} = 30$ and $NP_{late} = 10$.

4.2.26 PDcDE (Yu et al. 2022)

The authors in (Yu et al. 2022) split the population into two subpopulations: the first contained the best $\lfloor N/4 \rfloor$ individuals for exploitation, and the second population for exploration. The population size is adjusted using the proposed auto-controlled strategy. Two counters track consecutive improvements or stagnation of the best solution. When the optimization process is stable, and the best solution improves for L consecutive iterations, the worst individual is removed, thereby accelerating convergence. On the other hand, when stagnation is detected, and no improvement in the best solution occurs for L consecutive iterations, a randomly generated individual is added to increase diversity and help escape local minima. The population size is bounded by $NP_{max} = 18 \times D$ and $NP_{min} = 8 \times D$, while the authors set the threshold $L = 10$ in the experiments.

4.2.27 TDE (Meng and Yang 2022)

TDE, proposed by Meng and Yang (2022), utilizes a large fixed population initially and switches to a linear population size reduction strategy in the later stages of the search:

$$NP_{g+1} = \begin{cases} NP_g, & \text{if } nfes < p_s \times MaxFEs, \\ \frac{NP_{end} - NP_{init}}{(1 - p_s) MaxFEs} \times nfes + NP_{init}, & \text{otherwise.} \end{cases} \quad (66)$$

p_s is set to 0.05. The population size is kept constant during the first 5% of the search budget and then decreases linearly toward a predefined minimum population size for the remainder of the optimization process. In their experiments, $NP_{init} = 25 \times \log(D)\sqrt{D}$ and $NP_{end} = 4$.

4.2.28 FL-ADE (Zheng and Luo 2022)

An adaptive DE based on fitness landscape (FL-ADE), proposed by Zheng and Luo (2022), adapts the population size based on the fitness landscape.

$$NP_{g+1}^{FL} = \text{round} [(NP_{init} - NP_{min}) \times \phi + NP_{min}]. \quad (67)$$

ϕ is the local fitness landscape characteristic value. When ϕ is close to 0, it is close to the unimodal local fitness landscape, and NP is close to NP_{min} ; when ϕ is close to 1, it is close to the multi-modal local fitness landscape, and NP is close to NP_{init} . Readers interested in details on how ϕ is determined are advised to read the paper (Zheng and Luo 2022). FL-ADE alters the population size following the equation:

$$NP_{g+1} = \begin{cases} \text{round} \left(\frac{NP_{g+1}^{FL} + NP_{g+1}^{linear}}{2} \right), & \text{if } nfes < FES_t, \\ \min(NP_{g+1}^{FL}, NP_{g+1}^{linear}), & \text{otherwise.} \end{cases} \quad (68)$$

NP_{g+1}^{linear} is LPSR mechanism (see for example Eq. (38)), $FES_t = \frac{1}{2} \times MaxFES$, $NP_{init} = 25 \times \log(D)\sqrt{D}$, and $NP_{min} = 4$.

4.2.29 Mechanism PPSCM (Pereira de Lacerda et al. 2022)

Pereira de Lacerda et al. (2022) proposed a parameterless population-size control mechanism applicable to both evolutionary and swarm-based algorithms. We refer to this mechanism as PPSCM. The method dynamically increases or decreases the population according to the progress of the search. Population reduction is triggered when improvements in the global best solution are observed, promoting exploitation, whereas stagnation encourages population growth to enhance exploration. The adjustment is governed by a population-change velocity, and newly added individuals are generated around existing promising solutions while removed individuals are selected through a fitness-based procedure.

4.2.30 PRS (Li et al. 2023d)

Population reduction with individual similarity (PRS) (Li et al. 2023d) employs a linear differential decrease method to automatically determine the optimal population size for each generation.

$$NP_{g+1} = NP_{init} + (NP_{end} - NP_{init}) \times \left(\frac{nfes}{MaxFES} \right)^2. \quad (69)$$

The current population is divided into two subgroups of equal size according to individual similarity, which is computed using the cosine similarity:

$$IS_i = IS(\mathbf{x}_{i,g}, \mathbf{x}_{i+1,g}) = \frac{\mathbf{x}_{i,g} \times \mathbf{x}_{i+1,g}}{\|\mathbf{x}_{i,g}\| \times \|\mathbf{x}_{i+1,g}\|} = \frac{\sum_{j=1}^D x_{i,g}^j \times x_{i+1,g}^j}{\sqrt{\sum_{j=1}^D (x_{i,g}^j)^2} \times \sqrt{\sum_{j=1}^D (x_{i+1,g}^j)^2}}. \quad (70)$$

First, the similarity between each individual and the best individual $\mathbf{x}_{best,g}$ is computed. Individuals are sorted in descending order according to their similarities, and then split into two subgroups. Individuals in the first subgroup are more similar to the best individual than the individuals in the second subgroup. When population reduction is necessary, individuals are eliminated from the second subgroup based on their similarity to the best individual in this subgroup. After the elimination, the subgroups are merged into the current population that needs to evolve.

Li et al. (2023d) incorporated the PRS strategy into five basic DE and six advanced DE algorithms to evaluate the effect of PRS. In their experimental setup, the parameters related to population size reduction were set to $NP_{init} = 400$, and NP_{end} was assigned values between 4 and 7, depending on the mutation strategy employed.

4.2.31 Algorithm with Fluctuant Population (FP) strategy (Sun et al. 2023a)

The algorithm, proposed in (Sun et al. 2023a), uses the DE/best/1 mutation strategy. The population size in the next generation is highly variable during the optimization process. The individuals are divided into two groups: the current population (Group A) and the spare group (Group B). In the FP strategy, the parameter MP is calculated as follows:

$$MP = \frac{\left(\frac{NP_{init}}{D} - 1\right) \times (MaxFEs - nfes)}{MaxFEs} + 1.$$

and then NP_{g+1} is expressed as follows:

$$NP_{g+1} = \text{round} \left[(MP \times D - NP_{min}) \times 0.5 \times \left(\cos \left(\frac{nfes \times \pi}{MP \times D^2} \right) + 1 \right) \right] + NP_{min}. \quad (71)$$

Compared to the population size NP_g of the previous generation, if $NP_{g+1} < NP_g$, the worst $NP_g - NP_{g+1}$ individuals are removed from Group A and transferred to Group B. Conversely, if $NP_{g+1} > NP_g$, the best $NP_{g+1} - NP_g$ individuals are selected from Group B and added to Group A.

The parameters NP_{init} and NP_{min} were set to $5 \times D$ and 15, respectively.

4.2.32 HPDE (Meng and Zhang 2023b)

A slower decrease in population size during the later phases of evolution, together with an appropriately designed mutation strategy, can help the algorithm escape certain local optima. Motivated by this insight, Meng and Zhang (2023b) refined the population-reduction scheme used in PaDE as follows:

$$NP_{g+1} = \begin{cases} \left\lceil \frac{NP_{end} - NP_{init}}{(\frac{2}{3}MaxFEs - NP_{init})^2} \times (nfes - NP_{init})^2 + NP_{init} \right\rceil, & \text{if } nfes \leq 0.5 \times MaxFEs, \\ \left\lceil \frac{NP_{end} - \frac{1}{3}NP_{init}}{\frac{1}{3}MaxFEs} \times (nfes - MaxFEs) + NP_{end} \right\rceil, & \text{otherwise.} \end{cases} \quad (72)$$

The parameters were initially set to $NP_{init} = 18 \times D$ and $NP_{end} = 4$.

4.2.33 AGDE-MPP (Zhang et al. 2023a)

Zhang et al. (2023a) proposed Adaptive Guided Differential Evolution algorithm on Mutation, Parameter and Population (AGDE-MPP), which has improved the DE algorithm by adopting a new mutation scheme, a parameter adaptation scheme, and a non-linear population size reduction strategy. AGDE-MPP uses a hyperbolic tangent function in a non-linear population size reduction strategy:

$$NP_{g+1} = \frac{NP_{init} + NP_{end}}{2} + \frac{NP_{init} - NP_{min}}{2} \times \tan \left(-3 + 6 \times \frac{MaxFEs - nfes}{MaxFEs} \right). \quad (73)$$

The reduction strategy controls the population size according to the hyperbolic tangent function curve between $[-3, 6]$. In the early stage of the evolutionary process, the convergence speed of the population size function is slow. In the middle stage, the population size decreases linearly. In the later stage, the convergence speed of the curve slows again, so the population size is decreased non-linearly. To promote the exploration capability in the early stages of the evolutionary process, the population size is relatively large to ensure population diversification. In the later stages of the evolutionary process, the population size no longer decreases to find the optimum with as few individuals as possible. The population size of AGDE-MPP in the first three quarters of the evolutionary process is relatively large, helping avoid the solution from being trapped in a local optimum.

Additionally, the AGDE-MPP algorithm (Zhang et al. 2023a) sets different initial population sizes for different dimensions to avoid premature convergence. The final population size NP_{end} was 12 and the initial population size NP_{init} was $10 \times D$, $6 \times D$ and $4 \times D$ for the dimensions of 10, 30 and 50, respectively.

4.2.34 PE-SHADE (Zhang et al. 2023b)

An enhanced algorithm with a population entropy-based population adaptation strategy under the framework of SHADE (PE-SHADE) has already been proposed in a conference paper (Zhang et al. 2019). Work (Zhang et al. 2023b) presents a more comprehensive presentation and analysis of the experimental results.

4.2.35 MWADE (Tian et al. 2023)

In MWADE, proposed by Tian et al. (2023), the differences among individuals are used to assess the population's search requirements and dynamically adjust the population reduction rate. Specifically, at generation g , the population size is computed as follows:

$$NP_g = \text{round} \left((NP_{end} - NP_{init}) \times \left(\frac{nfes}{MaxFEs} \right)^{(1+e^{-\Delta f})} + NP_{init} \right), \quad (74)$$

$$\Delta f = \sum_{i=1}^{NP} \|\mathbf{x}_i^g - \mathbf{x}_{best}^g\|_2. \quad (75)$$

Δf denotes the sum of the Euclidean distances between all individuals $\mathbf{x}_i$ and the current best individual $\mathbf{x}_{best}$. A larger value of Δf indicates a higher level of population diversity, suggesting that the population still requires exploration; therefore, the population size is reduced more slowly. Conversely, a smaller value of Δf implies that the individuals are concentrated in a promising region of the search space. In this case, the population size is reduced more rapidly to accelerate convergence and enhance exploitation. In their experiments, the best results were obtained with $NP_{init} = 18 \times D$ and $NP_{end} = 10$.

4.2.36 EHMDE (Lv et al. 2023)

Exponential hybrid mutation differential evolution (EHMDE), proposed by Lv et al. (2023), utilizes an exponential population size reduction:

$$NP_{g+1} = \text{round} \left\{ (NP_{init} - NP_{end}) \times \exp \left[\frac{-nfes}{MaxFEs + 1 - nfes} \right] + NP_{end} \right\}. \quad (76)$$

In their experiments, NP_{end} was set to 10 and NP_{init} to 50. Note, in the original paper, the numerator is given as $MaxFEs$, which is likely a typographical error. We conjecture that it should instead be $-nfes$. With this correction, the population size decreases exponentially from 50 to 10 over the course of the search process.

4.2.37 EPSR (Ekinci et al. 2023)

SHADE algorithm with Exponential Population Size Reduction (EPSR) was used in a gravity inversion procedure to reconstruct the three-dimensional basement relief

geometry in sedimentary basins (Ekinici et al. 2023). Population size is calculated as follows:

$$NP_{g+1} = \text{round}[NP_{max} \times e^{-\lambda \times n_{fes}}]. \quad (77)$$

The parameter λ controls the exponential population size reduction. It is determined before the optimization process so that the population size decreases from NP_{init} to NP_{end} over the course of the search. In their experiments, the best results were obtained with $NP_{init} = 5 \times D$. NP_{end} was set to 4.

4.2.38 PaDE-pet (Meng 2023)

PaDE-pet, proposed by Meng (2023), uses the same population reduction as PaDE (see Section 4.2.7) from (Meng et al. 2019a). In the experiments, $NP_{init} = 25 \times \log(D)\sqrt{D}$ and $NP_{end} = 4$.

4.2.39 SHADE-cnEpSin (Hong et al. 2023)

In (Hong et al. 2023), the nonlinear population size reduction proposed by Stanovov et al. (2021) and presented in Section 4.2.17 was used. The resulting algorithm is called SHADE-cnEpSin.

4.2.40 APDSDE (Zhang et al. 2024)

Adaptive population size adjustment (APSA) method was proposed in a refined DE algorithm characterized by adaptive parameters and dual mutation strategies (APDSDE) (Zhang et al. 2024). This method maintains a larger population in the early iterations and a smaller population in later iterations. Population size for the next generation is calculated as:

$$NP_{g+1} = \text{round} \left(NP_{init} - (NP_{init} - NP_{end}) \left(\frac{n_{fes}}{MaxFEs} \right)^1 - \left(\frac{n_{fes}}{MaxFEs} \right)^2 \right). \quad (78)$$

In the experimental configuration adopted in (Zhang et al. 2024), the initial population size was set to $NP_{init} = 18 \times D$, while the final population size was set to $NP_{end} = 4$.

4.2.41 NLPSR-jSO (Zhou and Huang 2024)

Zhou and Huang (2024) proposed a variant of jSO algorithm with the NLPSR. It adaptively assigns more evaluation resources to dominant individuals, allowing their

potential to be fully exploited. The NLPSR method is defined as:

$$NP_{g+1} = \text{round} \left(NP_{init} + (NP_{end} - NP_{init}) \times \left(\frac{nfes}{MaxFEs} \right)^{k_1 \times e^{-k_2 \times \frac{nfes}{MaxFEs}}} \right), \quad (79)$$

where k_1 and k_2 are control parameters that balance exploration and exploitation performance throughout the evolutionary stage. The parameter k_1 determines the rate of population size reduction during the early evolutionary generations, whereas k_2 controls the rate of decrease in the late generations. The initial population size was set to $NP_{init} = 75 \times D^{\frac{2}{3}}$ and end population size to $NP_{end} = 4$.

NLPSR-jSO further incorporates a novel mechanism, known as Adaptive Archive Usage, which mitigates excessive archive utilization and enhances solution accuracy. Moreover, the mutation strategy is enhanced through a modified rank-based selective pressure, aimed at increasing population diversity and reducing the risk of premature convergence. Zhou and Huang (2024) experimentally showed that the cooperation of all three mechanisms improves the performance of jSO.

4.2.42 DDPSA framework (Qin et al. 2024)

In (Qin et al. 2024), a dimensional difference-based population size adjustment framework (DDPSA) was proposed. The framework monitors convergence speed by examining the dimensional differences between parent and offspring individuals. Based on the convergence speed information provided by the monitoring component, the mechanism adaptively regulates population size—both increasing and decreasing it—by leveraging the knowledge from the objective and solution spaces, thereby effectively mitigating premature convergence and evolutionary stagnation.

Table 5: Variation rules of population size in the FD-PSA strategy.

Indicator	Increase	Unchanged	Decrease
<i>fitStd</i>	-3	0	+3
<i>solutionStd</i>	-1	0	+1
<i>crowding</i>	-2	0	+2

When premature convergence occurs, the population size is expanded at a predefined rate p ($p = 0.01$). The number of individuals added each time is $\lceil p \times NP \rceil$. To effectively enhance population diversity, a dynamic opposition-based learning (DOL) strategy is employed to generate newly added individuals. In the DOL approach, the search space boundaries associated with each solution are adaptively adjusted. To ensure that newly generated individuals are positioned far from regions where the

current population is concentrated—and to increase the likelihood of exploring areas with greater optimization potential—the DOL strategy is applied to the top-ranked elite individuals according to fitness.

Analogous to the population expansion mechanism, the framework applies a population reduction strategy to mitigate stagnation, removing $\lceil p \times NP \rceil$ redundant individuals from the population. To prevent stagnation caused by low population size, a lower limit value NP_{limit} of the size NP has been set in the DDPSA framework:

$$NP_{limit} = (NP_{min} - NP_{init}) \times \left(\frac{n_{fes}}{MaxFes} \right) + NP_{init}. \quad (80)$$

Unlike traditional population reduction strategies that rely on fitness evaluation, the proposed approach introduces a new evaluation criterion that incorporates information from both the objective and solution spaces. In the experiments, $NP_{init} = 50$, $NP_{max} = 200$, and NP_{min} is set to the minimum size value to keep the mutation working properly.

4.2.43 Mechanism FD-PSA (Peng and Chen 2024)

In (Peng and Chen 2024), the population size is adaptively adjusted based on the trends of three metrics: the standard deviation of fitness values (*fitStd*), the standard deviation of solutions (*solutionStd*), and the crowding degree (*crowding*). For each metric, the change between consecutive iterations is evaluated and mapped to an increment or decrement in the population size, as shown in Table 5. The overall adjustment is obtained as the sum of contributions from all metrics. This results in a feedback-driven mechanism in which the population size increases when diversity decreases or convergence is detected, and decreases when sufficient diversity is present. Therefore, we refer to this strategy as the Feedback-Driven Population Size Adaptation (FD-PSA) strategy.

4.2.44 PSASIDEA (Yang et al. 2024b)

Yang et al. (2024b) proposed a Population Size Adaptive Strategy (PSASIDEA), in which the search state is monitored using population similarity, diversity, and evolution rate indicators. During the first half of the optimization process, the population size may be increased to improve diversity, whereas in the later stages it may be reduced depending on the fitness value. The population size is adjusted according to:

$$NP_g = \begin{cases} NP_{init} + (NP_{max} - NP_{init}) \times \frac{f(\mathbf{x})}{\max f(\mathbf{x})}, & g \in [0, 0.5G], \\ NP_{0.5G} - (NP_{0.5G} - NP_{min}) \times \frac{f(\mathbf{x})}{\max f(\mathbf{x})}, & g \in [0.5G, G]. \end{cases} \quad (81)$$

In the given equation, $\frac{f(\mathbf{x})}{\max f(\mathbf{x})}$ represents fitness-based scaling factor that controls the magnitude of population-size adaptation. G denotes the number of generations. They set it to 1000. $NP_{0.5G}$ denotes a population size after $G/2$ generations. In their experiments, NP_{init} was set to 100.

4.2.45 MHDE (Salgotra and Gandomi 2024)

In multi-hybrid DE (MHDE), proposed by Salgotra and Gandomi (2024), the population size is adjusted according to the improvement in the best fitness value between successive generations, ensuring that the reduction process is proportional to the algorithm's convergence behavior:

$$NP_{g+1} = \begin{cases} (1 - \Delta f_g^{\text{best}}) NP_g, & \text{if } \Delta f_g^{\text{best}} \leq \Delta f_{\text{max}}^{\text{best}}, \\ (1 - \Delta f_{\text{max}}^{\text{best}}) NP_g, & \text{if } \Delta f_g^{\text{best}} > \Delta f_{\text{max}}^{\text{best}}, \\ NP_{\text{min}}, & \text{if } NP_{g+1} < NP_{\text{min}}, \end{cases} \quad (82)$$

$$\Delta f_g^{\text{best}} = \frac{f_{g-1}^{\text{best}} - f_{g-2}^{\text{best}}}{|f_{g-2}^{\text{best}}|}. \quad (83)$$

$\Delta f_{\text{max}}^{\text{best}}$ is the threshold value.

4.2.46 APSDE (Cao et al. 2024)

Adaptive population size-based DE (APSDE), proposed by Cao et al. (2024), adapts the population size using historical information collected during the search. First, the similarity between the current population and archived population states is evaluated. The most similar records are then used to generate the new population size. The average population-size ratio is computed as a weighted Lehmer mean:

$$\mu_{NPR} = \frac{\sum_{i=1}^{|IRS|} w_i NPR_i^2}{\sum_{i=1}^{|IRS|} w_i NPR_i}. \quad (84)$$

where NPR_i denotes the ratio between the population size in the i -th archived state and the initial population size, while IR_i represents the corresponding improvement ratio of the best solution. IRS is the NPR_i set of the top $q\%$ similarity records. The weights are determined according to the historical improvement ratios:

$$w_i = \frac{IR_i}{\sum_{j=1}^{|IRS|} IR_j}. \quad (85)$$

The population size is then sampled from a normal distribution centered at μ_{NPR} :

$$NP = \text{round}(\mathcal{N}(\mu_{NPR}, 0.1) \times NP_{\text{init}}). \quad (86)$$

$\mathcal{N}$ is a random number generated from a normal distribution. If no improvement has been observed for several generations, the algorithm considers the population stagnated and adjusts the population size according to the search stage. During the first half of the search, the population is reduced to strengthen exploitation, whereas in the second half it is increased to restore diversity and improve exploration.

$$NP = \begin{cases} \text{round}\left(0.75 \times \frac{\sum_{j=1}^{|IRS|} IR_j}{|IRS|} \times NP_{init}\right), & \text{if } \frac{nfes}{MaxFEs} < 0.5, \\ \text{round}\left(1.25 \times \frac{\sum_{j=1}^{|IRS|} IR_j}{|IRS|} \times NP_{init}\right), & \text{otherwise.} \end{cases} \quad (87)$$

Additionally, during the final quarter of the search, the population is further reduced to intensify convergence:

$$NP = \begin{cases} \text{round}(\mathcal{N}(0.15, 0.1) \times NP_{init}), & \text{if } NP > \frac{NP_{init}}{4}, \\ NP, & \text{otherwise.} \end{cases} \quad (88)$$

In their experiments, NP_{min} is set to 4 and NP_{init} to $4 \times D$. The control parameter q is set to 0.3.

4.2.47 SHADE with CAPR (Sedelnikov et al. 2024)

Sedelnikov et al. (2024) employed SHADE with the CAPR mechanism proposed by Wong et al. (2017) (see Section 4.2.1) to optimize propeller geometry, aiming to minimize power consumption for a given thrust requirement. In their experiments, NP_{init} was set to 50 and 100, and NP_{min} to 10.

4.2.48 KBPDE (Liu et al. 2024)

KBPDE algorithm, proposed by Liu et al. (2024), combines adaptive Differential Evolution with K-modes clustering, BP neural network, and some other strategies for planning a globally optimal path for robots. It employs an exponential population size reduction strategy:

$$NP_g = NP_{init} \times e^{-t \times g} + NP_{end}, \quad (89)$$

where t is a random number between 0.1 and 0.2.

4.2.49 ACD-DE (Meng et al. 2024)

In (Meng et al. 2024), a hybrid parabolic linear population reduction is adopted in the proposed ACD-DE algorithm. For the first $\frac{2}{3}$ of the evolution progress, a parabolic linear population reduction is adopted for a slower reduction of the population size in the earlier stages of evolution to avoid premature convergence due to reduced exploration. After this stage, the linear population reduction is used. The population for each generation is calculated using the following equation:

$$NP_{g+1} = \begin{cases} \frac{y - NP_{init}}{(x - NP_{init})^2} \times (nfes - NP_{init})^2 + NP_{init}, & \text{if } nfes < x, \\ \frac{y - NP_{init}}{x - NP_{init}} \times (nfes - MaxFEs) + NP_{end}, & \text{otherwise,} \end{cases} \quad (90)$$

where $(x, y) = (\frac{2}{3} \times MaxFEs, \frac{1}{3} \times NP_{init})$ is the connecting point between the parabolic segment and the linear segment. In their experiments, $NP_{init} = 25 \times \log(D)\sqrt{D}$ and $NP_{end} = 4$. Similar reduction strategy is used in PaDE (see Section 4.2.7).

4.2.50 CEDE (Zeng et al. 2025)

Zeng et al. (2025) addressed the problem of premature convergence in the DE algorithm, which is mainly caused by the continuous reduction of population size and the resulting loss of population diversity. To enhance the exploration capability of DE, the authors propose a cosine-exponential population size adaptive (CEPSA) method.

CEPSA adaptively regulates the population size during the iterative process such that the overall trend is an exponential decrease, while the population size periodically increases according to a cosine function, as shown in Eq. (91). These periodic increases temporarily enhance population diversity, enabling the algorithm to escape local optima without explicitly detecting when stagnation occurs.

$$NP_{g+1} = NP_l^g \times (1 - \beta + \beta \times NP_c^g), \quad (91)$$

where

$$NP_l^g = NP_{init} \times 0.5^{(nfes/MaxFEs)} - 0.5 \times NP_{init} + NP_{end}$$

and

$$NP_c^g = |\cos(\theta \times g)|.$$

The described mechanism has two parameters, namely β , which determines the influence of the cosine term on population size adjustment, and θ , which controls the frequency of the cosine oscillation.

To maintain solution quality when the population size increases, CEPSA incorporates an archive of the best abandoned trial vectors. When the population expands, individuals are randomly selected from this archive and added to the population, after which they are removed from the archive to avoid repeated use. When the population size decreases, the worst individuals are removed from the population.

Authors recommend the initial population size of $NP_{init} = 28 \times D + 80 \times \sqrt{D} - 50$ and $NP_{end} = 4$. For parameters β and θ , the paper suggests using the following parameter values: $\beta = 0.05$ and $\theta = 0.04, 0.05, 0.07$, and 0.08 for $D = 10, 30, 50$, and 100 , respectively.

4.2.51 TSMS-DE (Xu et al. 2025)

The TSMS-DE (Xu et al. 2025) algorithm applies an opposition-based learning (OBL) strategy during population initialization. Initial individuals are first generated randomly within a predefined interval, and for each such individual, an opposite solution is computed using:

$$\mathbf{x}_{i,0}^{opp} = low_i + upp_i - \mathbf{x}_{i,0}, \quad (92)$$

where low_i and upp_i denote the lower and upper bounds of the i -th dimension. By generating opposite individuals and merging them with the original population, the algorithm improves search space coverage and enhances convergence speed. In addition, TSMS-DE incorporates a population diversity enhancement mechanism to prevent stagnation in later stages of the optimization process.

Population diversity is monitored using the relative diversity parameter:

$$R_{DP} = \frac{DP}{DP_{ini}}, \quad (93)$$

$$DP = \sqrt{\sum_i^{NP} \|\mathbf{x}_i - \bar{\mathbf{x}}\|^2}, \quad (94)$$

where DP denotes the current population diversity and DP_{ini} the initial diversity. The diversity is calculated, as shown in Eq. (94). If the diversity falls below a threshold ξ and an individual shows no improvement for N consecutive generations, the individual is considered stagnant. To reactivate such individuals, a new candidate solution is generated as follows:

$$\mathbf{x}_{i,g+1} = \mathbf{x}_{r_1,g} + F_c \times (\mathbf{x}_{r_2,g} - \mathbf{x}_{r_3,g}), \quad (95)$$

where $\mathbf{x}_{r_1,g}$, $\mathbf{x}_{r_2,g}$, and $\mathbf{x}_{r_3,g}$ are distinct randomly selected individuals, and F_c is a scaling factor drawn from a Cauchy distribution $\mathcal{C}(0.5, 0.1)$.

Additionally, TSMS-DE employs a parabolic-linear population size reduction scheme (see Eq. (42)) from the PaDE algorithm, which gradually decreases the population size to enhance search efficiency during the evolutionary process. The initial population size was $NP_{init} = 25 \times \log(D) \times \sqrt{D}$, while the final population size was $NP_{end} = 4$.

4.2.52 SL-SHADE (Wu et al. 2025)

In (Wu et al. 2025), the authors proposed the NLPSR strategy, which maintains a population size reduction similar to LPSR at an early stage of optimization. This allows the algorithm to perform the global search. In the middle and late stages of optimization, the population size reduction is more aggressive, and the optimization is guided to perform local searches. The described population size reduction mechanism is modeled by Eq. (96), where λ increases from 0 to 1 with generations g , so the population size decreases exponentially. The control parameteris NP_{init} and NP_{end} were set to $18 \times D$ and 4, respectively.

$$NP_{g+1} = \text{round}[(NP_{end} - NP_{init})(1 - e^\lambda) + NP_{init}]$$

$$\lambda = 1 - \frac{g_{\max}}{g_{\max} + 1 - g} \quad (96)$$

4.2.53 NLAPSMjSO-EDA (Shen et al. 2025)

In (Shen et al. 2025), the authors proposed the NLAPSMjSO-EDA algorithm with the NLPSR strategy that was taken from the NL-SHADE-RSP (Stanovov et al. 2021) algorithm, as described in Section 4.2.17. The NP_{init} parameter was analyzed and results demonstrate that the $NP_{init} = 150 \times D^{\frac{2}{3}}$ reaches the highest performance ranking. In this analysis, the NP_{end} parameter was set to 4.

4.2.54 DF-NPAS (Sun et al. 2025)

Sun et al. (2025) proposed a population adjustment strategy that is based on both *population diversity* and *fitness improvement*. We refer to this strategy as the

Diversity–Fitness Nonlinear Population Adjustment Strategy (DF-NPAS). The population diversity is used in the early stage of optimization, while the fitness improvement is used in the late stage. Additionally, a stagnation prevention mechanism is used to help the algorithm escape from local optima. The details of the strategy are as follows:

- **Early stage of optimization:** The population size is dynamically adjusted based on the relative population diversity, which is calculated as:

$$rDiv = \frac{Div}{Div_{init}}, \quad (97)$$

where Div and Div_{init} represent diversity of the current and initial population. Diversity is measured using the infinite norm with respect to the best individual:

$$Div = \frac{1}{NP} \sum_{i=1}^{NP} \|X_{i,g} - X_{best,g}\|_{\infty}. \quad (98)$$

The population size is updated using a nonlinear function:

$$NP_{g+1} = NP_g - u(rDiv - e^{-(z-0.1)}) + u(e^{-(z+0.1)} - rDiv), \quad (99)$$

$$z = \frac{nfes}{MaxFEs}; \quad u(x) = \begin{cases} 1, & x \geq 0, \\ 0, & x < 0. \end{cases}$$

According to this strategy, if value of $rDiv$ is greater than or equal to $e^{-(z-0.1)}$, an individual with the worst fitness value is deleted from the current population and if its value is less than $e^{-(z+0.1)}$, an individual is randomly selected from the external archive and added to the current population.

- **Late stage of optimization:** When $nfes$ is greater than $tfes$ (the stage threshold, good value is $MaxFEs/5$), the population size is adjusted based on fitness improvement:
 - if the best fitness continuously improves ($f_{best,g+1} < f_{best,g}$), the population size remains unchanged,
 - otherwise, it is updated according to:

$$NP_{g+1} = \text{round}(N_t \times \max(0.9, \sin(f \times Iter))), \quad (100)$$

$$N_t = (NP_{max} - NP_{min}) \times R_t + NP_{min}, \quad (101)$$

$$R_t = 1 - \frac{1}{1 + e^{10-15z}}, \quad (102)$$

where round means round to nearest integer, $Iter$ is the current iteration number, $\sin(\cdot)$ is a sine function, and f is the period of the sine function. When the NP_{g+1} is smaller than the NP_g , individuals with poor fitness are deleted from the current population, otherwise, $NP_{g+1} - NP_g$ individuals are randomly selected from the external archive and added to the current population.

- **Stagnation prevention:** When the population diversity decreases continuously or the fitness does not improve continuously, it indicates that the algorithm may fall into a local optimum or stagnate. To prevent stagnation, two global variables were introduced:

- CD – counts consecutive iterations of decreasing diversity,
- CF – counts consecutive iterations without fitness improvement.

When $CD \geq N_g$ (threshold, good value is 20) or $CF \geq N_g$, additional individual is randomly selected from external archive and inserted into the current population to help the algorithm escape from stagnation.

4.2.55 SAEDE-EP'25 (Budiman et al. 2025)

Self-Adaptive Ensemble-based DE with Enhanced Population Sizing (SAEDE-EP) was first published in (Budiman et al. 2020), and it is described in Section 4.2.11. In (Budiman et al. 2025), more detailed analysis and validation is given.

The SAEDE-EP algorithm was used in (Wang et al. 2025b), where the authors performed a sensitivity analysis of the population growth rate parameter that determines the addition or removal of individuals from the population.

4.2.56 E-ADE (Zhang et al. 2026)

Zhang et al. (2026) proposed an adaptive population size adjustment strategy within the enhanced adaptive differential evolution algorithm (E-ADE), in which the population size is regulated according to the variation of the global best fitness. When the improvement between consecutive generations exceeds a predefined threshold ($\Delta f > \epsilon$), a fraction of the population is removed:

$$N_{\text{dec}} = \lfloor \beta \times NP_g \rfloor. \quad (103)$$

N_{dec} is the number of individuals to be removed and β is the scaling factor, typically in the range $[0.1, 0.3]$.

4.3 Summary

The algorithms reviewed in Sections 2.3, and 4 can be categorized according to the type of population size adjustment they employ. Tables 6 and 7 provide an overview of these algorithms, where, in addition to the algorithm name, author(s), and publication year, the type of population size adjustment employed is indicated. For each algorithm, we have the following information:

- LPSR – Linear population size reduction that follows a linear trajectory throughout the evolutionary process.
- non-LPSR – Non-linear population size reduction that follows a predefined non-linear trajectory throughout the evolutionary process.
- Heuristic – The population reduction scheme dynamically determines the desired population size using additional information about the optimization process (e.g., diversity, fitness values, stagnation indicators, etc.).

Table 6: Algorithms and properties of their population size adjustment strategies (part 1).

Algorithm	Author(s) and year	LPSR	non-LPSR	Heuristic	Reduce only	Reduce+Increase
DESAP	Teo (2006)		✓			✓
dynNP-DE	Brest and Sepesy Maučec (2008)		✓		✓	
FDSADE	Tirronen and Neri (2009)		✓			✓
ADE	Zhabitsky and Zhabitskaya (2013)					✓ ¹
L-SHADE	Tanabe and Fukunaga (2014)	✓			✓	
jSO	Brest et al. (2017)	✓			✓	
CAPR	Wong et al. (2017)		✓		✓	
IDEbd	Bujok and Tvrdik (2017)		✓	✓		✓
EsDEr-NR	Awad et al. (2018)		✓		✓	
dyn-NP in LSGO	Maucec et al. (2018)		✓		✓	
EAGDE	Mohamed and Mohamed (2019)		✓		✓	
DMPsa mechanism	Poláková et al. (2019)	✓				✓
HARD-DE	Meng and Pan (2019)	✓	✓		✓	
PaDE	Meng et al. (2019a)	✓	✓		✓	
ESHADE_USM	Kumar et al. (2019)		✓		✓	
PE-SHADE'19	Zhang et al. (2019)		✓			✓
eL-SHADE	Khan et al. (2020)		✓		✓	
SAEDE-EP	Budiman et al. (2020)		✓	✓		✓
APDE	Zhang et al. (2020)		✓	✓		✓
PaDE-NPC	Meng et al. (2020b)	✓	✓		✓	
DE-NPC	Meng et al. (2020a)	✓	✓		✓	
ADDE	Zhan et al. (2020)		✓	✓		✓
FADE	Xia et al. (2021)		✓			✓
NL-SHADE-RSP	Stanovov et al. (2021)		✓		✓	
CS-DE	Meng et al. (2021)	✓	✓		✓	
Crowding dynNP-DE	Zhong et al. (2021)		✓		✓	
Hip-DE	Meng and Yang (2021)		✓		✓	
SLDE	Zeng et al. (2022)		✓ ²			✓
DPSHADE	Li et al. (2022b)	✓	✓		✓	
ExR, LoR and PaR	Najjarro et al. (2022)		✓		✓	
NL-SHADE-RSP-MID	Biedrzycki et al. (2022)		✓		✓	
IPM-DE	Morales-Castaneda et al. (2022)		✓	✓		✓
PDcDE	Yu et al. (2022)		✓	✓		✓
TDE	Meng and Yang (2022)		✓		✓	
FL-ADE	Zheng and Luo (2022)		✓	✓		✓
Mechanism PPSCM	Pereira de Lacerda et al. (2022)		✓	✓		✓
PRS	Li et al. (2023d)		✓		✓	
MjSO	Li et al. (2023b)	✓			✓	
FP	Sun et al. (2023a)		✓			✓
APSM-jSO	Li et al. (2023c)	✓			✓	
HPDE	Meng and Zhang (2023b)	✓	✓		✓	
cDE-APAR	Sun et al. (2023b)	✓			✓	
AGDE-MPP	Zhang et al. (2023a)		✓		✓	
PE-SHADE	Zhang et al. (2023b)		✓			✓
MWADE	Tian et al. (2023)		✓		✓	

¹Population is only increased in ADE.²SLDE is based on LPSR, but it uses a sawtooth function, so it can be classified as non-LPSR.

Table 7: Algorithms and properties of their population size adjustment strategies (part 2).

Algorithm	Author(s) and year	LPSR	non-LPSR	Heuristic	Reduce only	Reduce+Increase
EHMDE	Lv et al. (2023)		✓		✓	
EPSR	Ekinici et al. (2023)		✓		✓	
PaDE-pet	Meng (2023)	✓	✓		✓	
SHADE-cnEpSin	Hong et al. (2023)		✓		✓	
L-SRTDE	Stanovov and Semenkin (2024c)	✓			✓	
APDSDE	Zhang et al. (2024)		✓		✓	
NLPSR-jSO	Zhou and Huang (2024)		✓		✓	
DDPSA framework	Qin et al. (2024)		✓			✓
Mechanism FD-PSA	Peng and Chen (2024)		✓	✓		✓
PSASIDEA	Yang et al. (2024b)		✓	✓		✓
MHDE	Salgotra and Gandomi (2024)		✓	✓	✓	
APSDE	Cao et al. (2024)		✓	✓		✓
SHADE with CAPR	Sedelnikov et al. (2024)		✓		✓	
KBPDE	Liu et al. (2024)		✓		✓	
ACD-DE	Meng et al. (2024)	✓	✓		✓	
DSDCDE	Deng et al. (2025)	✓				✓
CEDE	Zeng et al. (2025)		✓			✓
FODE	Wang et al. (2025a)	✓			✓	
mLSHADE-SPACMA	Fu et al. (2025)	✓			✓	
TPADE	Gong et al. (2025)	✓			✓	
L-SRTDEwPSM	Popič et al. (2025)	✓			✓	
TSMS-DE	Xu et al. (2025)		✓		✓	
SL-SHADE	Wu et al. (2025)		✓		✓	
NLAPSMjSO-EDA	Shen et al. (2025)		✓		✓	
DF-NPAS	Sun et al. (2025)		✓	✓		✓
SAEDE-EP'25	Budiman et al. (2025)		✓	✓		✓
E-ADE	Zhang et al. (2026)		✓	✓	✓	

Table 8: Number of algorithms in this review by population size adjustment strategy characteristics.

	LPSR	non-LPSR	Heuristic	Reduce only	Reduce+Increase
Before 2017		4		2	3
Since 2017	21	56	15	45	22
Total	21	60	15	47	25

- Reduce only – Only population size reductions occur during the evolutionary process.
- Reduce+Increase – Both population size reductions and population size increases occur during the evolutionary process.

For clarity, in Tables 6 and 7, the reviewed algorithms are organized chronologically by year of publication. An algorithm may use a linear population size reduction (LPSR) strategy, a non-linear population size reduction (NLPSR) strategy, or a combination of multiple adjustment mechanisms. In some cases, algorithms combine LPSR and NLPSR, while others employ more than one NLPSR strategy. Furthermore, NLPSR strategies may incorporate additional heuristics based on information gathered during the optimization process, such as population diversity, fitness values, or stagnation indicators. Depending on the design of the adjustment mechanism, the population size may decrease monotonically throughout the search or be allowed to both decrease and increase adaptively. Table 8 shows the number of reviewed algorithms in each group, aggregated from Tables 6 and 7, before 2017, since 2017, and total. Since 2017, there have been 45 algorithms that utilized only reduction in population size, and 22 algorithms that combined both reduction and expansion of the population size.

In the next section, we present the common properties of the two groups of algorithms. The first group consists of algorithms that utilize heuristic methods for population size adjustment, whereas the second group comprises algorithms that apply a combination of two population size adjustment strategies during the evolutionary process.

5 Common properties of population size adjustment strategies

In this section, we summarize the common properties of population size adjustment strategies in DE-based algorithms, which were presented in Section 4. Population size adjustment encompasses a broad range of mechanisms, including learning-based population size control strategies, heuristic-based approaches, hybrid methods that combine multiple strategies, and related techniques.

As mentioned at the beginning of Section 4, the classification of algorithms solely into linear and non-linear strategies is neither trivial nor straightforward, as there is often a big overlap between used strategies. Therefore, based on these facts, we propose a new taxonomy that groups these population size adjustment strategies into the following categories:

- **Fixed** – The population size remains unchanged throughout the evolution process and is set before the evolutionary process starts. It is often calculated based on the problem characteristics, such as problem dimension. One advantage of a fixed population size is that it generally enables easier parallelization.
- **Deterministic control** – During the evolutionary process, the population size is adjusted based on a predefined trajectory, typically expressed using a linear or non-linear mathematical equation. These mechanisms often introduce a new set of parameters that need to be adjusted (e.g., start and end population sizes, curve/trajectory parameters, ...), but they may be less sensitive than other algorithm parameters. These mechanisms can be further grouped based on their chosen trajectory:

- **Linear trajectory** – Population reduction follows a linear trajectory (usually decreasing) from the initial population size to the final population size.
- **Non-linear trajectory** – Population reduction follows a predefined non-linear trajectory (e.g., parabolic or exponential) from the initial population size to the final population size.
- **Fluctuating trajectory** – Population size varies during the evolutionary process according to a predefined function, with both decreases and increases in the population size.
- **Heuristic** – This category could also be called *adaptive*. Population size change is dependent on the selected heuristic measure of the evolutionary process. These heuristics usually rely on measures such as population/individual fitness values, population diversity, and stagnation detection, enabling them to dynamically adapt the population size to each individual problem’s characteristics. Because they require additional knowledge that needs to be obtained during the evolutionary process, they often introduce additional computational overhead, particularly when population diversity measures are used.
- **Self-adaptive** – Population size is determined via additional information obtained from the evolutionary process. Unlike heuristic approaches, where the mechanism itself is predetermined and usually remains unchanged, self-adaptive mechanisms, which also include learning-based approaches, typically use feedback from the evolutionary process to adjust their population-size behavior. An example of a self-adaptive population size method is presented in (Teo 2006). These approaches can better adapt to the characteristics of the problem, uncover non-trivial relations between population size and the problem, and may reduce the need to manually set and tune the mechanism’s behavior. Their potential downside is that they may introduce additional parameters, intricate mechanisms, and substantial computational overhead.
- **Hybrids** – These approaches combine two or more aforementioned strategies either by applying them sequentially or by integrating them into a single population-size adjustment mechanism. Typically, this is accomplished by using a predetermined point in the evolutionary process at which population size adjustment strategies are switched (see (Meng and Pan 2019; Meng et al. 2019a; Li et al. 2022b), etc.). Hybrid approaches aim to exploit the advantages of multiple strategies, but they introduce a new problem of determining a suitable switching point (often called a pivot).

In addition to deterministic strategies, the reviewed algorithms frequently used heuristic and hybrid strategies. Given their prevalence and the greater variability in how population size adjustment is implemented within these strategies, we study them in greater detail in the following two subsections.

5.1 Heuristic strategies

Some of the algorithms analyzed in Section 4.2 utilized additional knowledge about the population, the optimization problem’s landscape, the progress of evolution, or a combination of these. Although they all belong to a group of non-linear population

Table 9: Algorithms with the heuristic population size adjustment strategies.

Algorithm	Description
IDEbd	Population size adapts to current diversity , with a linearly decreasing diversity target.
SAEDE-EP	Population changes only upon best-solution stagnation , otherwise, population size is unchanged.
APDE	When the best solution stagnates , inferior individuals may enter into population, and ineffective individuals are periodically pruned based on fitness and stagnation history.
ADDE	The population size is dynamically reduced when the best solution shows significant improvement, and increased when improvement is poor to enhance diversity , using an adaptive threshold that decreases over time.
IPM-DE	The population size is increased when population diversity stagnates , and reduced when diversity falls below a threshold of the maximum diversity ever observed.
PDcDE	The population size is reduced by removing the worst individual when the best solution improves consistently, and increased by adding a random individual when stagnation occurs.
FL-ADE	The population size is adjusted adaptively in each generation based on the fitness landscape . It remains larger in multimodal landscapes and shrinks in unimodal landscapes to speed up convergence.
Mechanism PPSCM	The population size is controlled by a velocity that increases during stagnation and decreases during improvement. Individuals are added near good solutions or removed. Changes taper off near the end to reduce noise.
Mechanism FD-PSA	The population size is adjusted dynamically based on three metrics: fitness standard deviation , solution-space standard deviation , and crowding degree .
PSASIDEA	The adaptive strategy monitors search state using diversity and evolution indicators. The population size may be increased during the first half of optimization to enhance diversity, then decreased in the later stages based on fitness performance .
MHDE	Uses an adaptive proportional reduction mechanism that shrinks the population size over time based on improvements in the best fitness value between successive generations.
APsDE	The population size is adopted using historical search information and similarity between current and archived population states. When progress stagnates, the algorithm further adjusts population size based on the search stage, reducing it early for exploitation and increasing it later to maintain diversity, with an additional reduction near the end for faster convergence.
DF-NPAS	The population size is controlled using a dynamic, two-stage nonlinear strategy that prioritizes population diversity in early iterations and fitness improvement in later stages.
SAEDE-EP'25	The population size is adjusted whenever stagnation is detected. A growth rate determines the new population size and is selected from a set of possible growth rates or an archive of successful ones.
E-ADE	The population size is adaptively reduced during optimization according to the variation of the global best fitness improvement , with larger population maintained early for exploration and smaller ones later for convergence. Only low-performing individuals are removed.

size adjustment strategies, they do not explicitly follow a predefined reduction curve. We will refer to these strategies as heuristic strategies for population size adjustment.

Table 9 shows all 15 algorithms that use heuristics for population size adjustment. Note that all these algorithms belong to a group of non-linear population size adjustment strategies. The NLPSR group of algorithms are collected in Tables 6 and 7. In Table 9, we have highlighted the keywords that describe the population size adjustment strategies used by the algorithms. The following keywords appear most often:

- diversity,
- best solution and/or best fitness,
- stagnation, and
- fitness improvement.

With these 15 algorithms that use heuristics for population size adjustment, we can observe the growth of the complexity of mechanisms over the years, which is reflected by the increasing number of mechanisms, which are also more sophisticated.

5.2 Hybrid strategies

To leverage the advantages of both linear and non-linear population reduction strategies, 10 algorithms presented in Section 4.2 employ a hybrid approach that combines the two. Table 10 summarizes these algorithms, together with brief descriptions and the equations defining the non-linear component of each strategy. A common characteristic among these approaches is the application of a non-linear reduction scheme during the early stages of the optimization process, followed by a linear reduction strategy in the later stages. This design aims to balance the exploratory properties of non-linear reduction with the stable convergence behavior associated with linear population reduction.

6 Experiment on CEC 2024 benchmark suite

In this section, the benchmark suite is presented, followed by experimental setup, results, and discussion. Simulation results comparing linear population size reduction with selected state-of-the-art non-linear population size reductions/adjustments are presented.

6.1 Benchmark suite

The experimental setup follows the rules and evaluation procedure specified by the CEC 2024 Special Session and Competition on Single-Objective Real-Parameter Numerical Optimization (Qiao et al. 2023), which builds upon the CEC 2017 benchmark suite (Awad et al. 2016). It consists of 29 shifted, rotated, non-separable, and ill-conditioned test functions designed to reflect the complexity of real-world optimization problems. The benchmark includes unimodal functions, simple multimodal functions, hybrid functions, and composition functions, each posing increasing levels of difficulty for optimization algorithms. The functions in the four given categories are:

Table 10: Algorithms that combine two population size adjustment strategies.

Algorithm	Combined strategies	Equation of nonlinear part
HARD-DE	Parabolic until $\frac{2}{3} \times \text{MaxFEs}$, then Linear	$\left[\frac{y - NP_{init}}{(x - NP_{init})^2} \times (nfes - NP_{init})^2 + NP_{init} \right]$
PaDE	Parabolic until $\frac{2}{3} \times \text{MaxFEs}$, then Linear	$\left[\frac{y - NP_{init}}{(x - NP_{init})^2} \times (nfes - NP_{init})^2 + NP_{init} \right]$
PaDE-NPC	Fixed population size until $0.15 \times \text{MaxFEs}$, then Linear	constant population size
DE-NPC	Parabolic until $\frac{2}{3} \times \text{MaxFEs}$, then Linear	$\left[\frac{y - NP_{init}}{(x - NP_{init})^2} \times (nfes - NP_{init})^2 + NP_{init} \right]$
CS-DE	Elliptic until $\frac{2}{3} \times \text{MaxFEs}$ then Linear	$\left[\sqrt{NP_{init}^2 - \frac{NP_{init}^2 - y^2}{x^2}} \times nfes^2 \right]$
DPSHADE	Nonlinear until $\frac{1}{2} \times \text{MaxFEs}$ then Linear	$\text{round}((NP_{end} + NP_{init}) \times M + NP_{init}),$ $M = \frac{1}{1 + \exp\left(-\frac{10 \times nfes}{\text{MaxFEs}} + 5\right)}$
FL-ADE	Nonlinear until $\frac{1}{2} \times \text{MaxFEs}$ then Linear	See Eqs. (67) and (68)
HPDE	Nonlinear until $\frac{1}{2} \times \text{MaxFEs}$ then Linear	$\left[\frac{NP_{end} - NP_{init}}{(\frac{2}{3} \text{MaxFEs} - NP_{init})^2} \times (nfes - NP_{init})^2 + NP_{init} \right]$
PaDE-pet	Parabolic until $\frac{2}{3} \times \text{MaxFEs}$, then Linear	$\left[\frac{y - NP_{init}}{(x - NP_{init})^2} \times (nfes - NP_{init})^2 + NP_{init} \right]$
ACD-DE	Parabolic until $\frac{2}{3} \times \text{MaxFEs}$, then Linear	$\left[\frac{y - NP_{init}}{(x - NP_{init})^2} \times (nfes - NP_{init})^2 + NP_{init} \right]$

- *unimodal functions* (f1, f2): the Bent Cigar function and the Zakharov function. These problems are conceptually the simplest and are expected to pose only moderate difficulty for well-designed optimizers.
- *multimodal functions* (f3 – f9): Rosenbrock’s, Rastrigin’s, Expanded Schaffer’s F6, Lunacek Bi-Rastrigin, Non-continuous Rastrigin, Levy, and Schwefel. Although these functions contain multiple local optima and are both shifted and rotated, their landscapes typically retain a relatively regular structure, allowing many algorithms to explore them effectively.

- *hybrid functions* (f10 – f19): functions constructed by combining several basic functions. Because each component contributes differently across regions of the search space, the resulting landscapes vary substantially from one region to another.
- *composition functions* (f20 – f29): functions, defined as weighted combinations of basic functions with an added bias that determines which component optimum becomes global. These are generally regarded as the most difficult, as they integrate diverse landscape characteristics and hybrid components as well.

The search domain for all functions is $[-100, 100]^D$, and the problems are treated as black-box functions. The maximum number of function evaluations *MaxFEs* is $10000 \times D$, where D is problem dimensionality.

6.2 Experimental setup

In the experiments, the jSO algorithm was used as the baseline, because it has been widely used as a foundation for many improved DE variants and has received significant attention in the literature. For this reason, we considered it an appropriate and representative modern DE framework for studying population size reduction strategies.

Various population-size adjustment strategies—originally proposed within different DE-based frameworks—were integrated into the jSO algorithm. This unified implementation enables clearer comparisons and assessment of each strategy’s impact, since other algorithm-specific mechanisms do not confound performance. A strategy was eligible for the experimental comparison if:

- It defines a distinct population-size adjustment mechanism — strategy distinctiveness.
- The population-size adjustment mechanism can be isolated from the original algorithm and implemented within jSO without transferring other algorithm-specific components — transferability.
- Its behavior can be expressed under the common experimental setting — parameter compatibility.
- The selected set covers the principal mechanism categories identified by the review — mechanism coverage.
- Within conceptually similar mechanisms, one representative strategy is selected rather than testing multiple near-duplicates — a redundancy.

Based on these criteria, the 15 population size adjustment strategies were selected to represent a broad range of population-size adjustment principles while avoiding unnecessary duplication. The original jSO algorithm was retained as the baseline, representing the widely used LPSR approach. The strategy of dynNP-DE algorithm was included as one of the earlier population-size reduction approaches, providing a historical counterpart to more recent strategies. The strategy of the APDSDE algorithm represents a simple non-linear reduction rule without additional control parameters, whereas that of the NLPSR algorithm provides a more general parameterized exponential reduction model. The strategy NLPSR-ver2 was additionally considered to

investigate how different parameter settings of the same general reduction model affect performance.

Several strategies were selected to represent adjustment mechanisms that combine different reduction principles or different stages of the evolutionary process. The population size reduction strategy in the HPDE algorithm combines non-linear and linear reduction over two equally sized stages, whereas those in HARD-DE and CS-DE algorithms combine linear reduction with parabolic and elliptic reduction, respectively, with the transition occurring at $\frac{2}{3} \times \text{MaxFEs}$. The strategy in the DPSHADE algorithm provides another distinct combination of non-linear and linear reduction, using a sigmoid-shaped reduction during the first half of the evolutionary process, followed by linear reduction. These strategies, therefore, enable comparison of different shapes and transition points in stage-dependent population-size adjustment.

The selection also includes strategies with substantially different population-size trajectories. The strategy of the AGDE-MPP algorithm uses a hyperbolic-tangent reduction curve, while those of the EAGDE and APDSDE algorithms represent non-linear trajectories that reduce the population more rapidly than LPSR during parts of the evolutionary process (see Figure 2). In contrast, PRS strategy follows a slower reduction trajectory and additionally incorporates information on individual similarity. FP strategy was selected because it introduces pronounced fluctuations in population size (see Figure 3), whereas the SLDE algorithm employs a sawtooth-linear mechanism that allows both population reduction and increase. These approaches extend the comparison beyond monotonically decreasing population-size trajectories.

Finally, the strategy of DSDCDE algorithm was included to represent a search-state-dependent approach, in which population-size adjustment exploits information from both the fitness and search spaces. Together, the selected strategies cover linear and non-linear reduction, parameterized and parameter-free approaches, monotonic and fluctuating population-size trajectories, combinations of multiple reduction schemes, reduction-and-increase mechanisms, and approaches that exploit information obtained during the evolutionary process. Thus, the selected set is intended to provide broad coverage of qualitatively different population-size adjustment principles rather than an exhaustive comparison of all strategies identified in the review.

Table 11 gives a collection of the algorithms used in the experiment. The first algorithm is jSO (Brest et al. 2017). It is used as a framework which includes one of the selected population size adjustment strategies and combines it with the jSO algorithm. If, for example, the fluctuant population (FP) size adjustment strategy is incorporated into the jSO, we denote this algorithm as jSO+FP. Note that each variant in the jSO algorithm incorporates only the population size reduction adjustment strategy and not the entire original algorithm that proposed the strategy. The original algorithm may include other mechanisms, such as F and CR parameters adaptation, sub-populations, etc.

All algorithms listed in Table 11 have the same initial and final (minimum) population sizes, as those in the jSO algorithm, namely $NP_{init} = 25 \times \log(D)\sqrt{D}$ and $NP_{end} = 4$.

All other parameters of each population size reduction/adjustment strategy used in the jSO+*strategy* algorithm remain the same as proposed in the original algorithm.

Table 11: Compared population size adjustment strategies applied in the jSO algorithm.

Algorithm	Strategy's description
jSO+APDSDE	adaptive population size adjustment (APSA) (see Sect. 4.2.40)
jSO+dynNP-DE	dynamic population size reduction $diffPS = 4$ (dynNP-DE) (see Sect. 2.3.2)
jSO+NLPSR	non-linear population size reduction (NLPSR) (see Sect. 4.2.41)
jSO+NLPSR-ver2	non-linear population size reduction (NLPSR) with $k_1 = 3.0, k_2 = 1.5$ (see Sect. 4.2.41)
jSO+DSDCDE	dual-space-based population size adaptive method (DSPSA) (see Sect. 4.1.6)
jSO+FP	fluctuant population (FP) (see Sect. 4.2.31)
jSO+HPDE	dynamic hierarchical population (see Sect. 4.2.32)
jSO+HARD-DE	parabolic population size reduction (see Sect. 4.2.6)
jSO+CS-DE	combined elliptic and linear reduction approach (see Sect. 4.2.18)
jSO+DPSHADE	combined linear and non-linear reduction approach (see Sect. 4.2.22)
jSO+AGDE-MPP	NLPSR, according to the hyperbolic tangent function curve (see Sect. 4.2.33)
jSO+SLDE	sawtooth-linear population size adaptive method (see Sect. 4.2.21)
jSO+PRS	population reduction with individual similarity (PRS) (see Sect. 4.2.30)
jSO+EAGDE	NLPSR, according to $X = 1 - (nfs / MaxFEs)$ (see Sect. 4.2.5)
jSO	linear population size reduction (LPSR) (see Sect. 2.2)

The jSO+NLPSR-ver2 algorithm differs from the jSO+NLPSR algorithm only in the setting of two population size reduction parameters. The jSO+NLPSR algorithm applies:

$$k_1 = 1.6 \wedge k_2 = 2.0,$$

while the jSO+NLPSR-ver2 algorithm uses:

$$k_1 = 3.0 \wedge k_2 = 1.5.$$

Experiments were conducted for problem dimensions $D = 10, 30, 50$, and 100 . For each test function, every algorithm was independently executed 25 times. This number of runs allows reliable estimation of both central tendency (e.g., mean or median) and variability (e.g., variance), and provides adequate statistical power for common significance tests. Each run was terminated either when the obtained error fell below 10^{-8} , or when the maximum number of function evaluations has reached $MaxFEs$.

6.3 Results

In this section, we present the experimental results. We first examine the behavior of the proposed strategies, focusing on how they reduce and/or increase the population size throughout the evolutionary process. This is followed by a statistical comparison of the strategies using the Wilcoxon signed-rank test and the Friedman test. In addition, convergence graphs for selected benchmark problems are presented and discussed to further illustrate the performance characteristics of the algorithms.

6.3.1 Population size variations

The variation curves of population size with the number of function evaluations for 10D are shown in Figure 2. Figure 3 depicts the variation curves for the FP strategy and jSO. Compared with the linear population reduction strategy of jSO and DSDCDE, the population sizes of other strategies varied a lot in the first half of the evolutionary process. The strategies are trying to avoid getting trapped in the local optimal solution. In order to promote the exploration capability in the early stages of evolutionary process, the strategies employ a relatively large population size to ensure population diversification. In the mutation operation, DE employs a large number of individuals. With the progress of evolutionary process, some individuals with poor fitness function values are deleted. The mutation, crossover and selection operations gradually strengthen the exploitation ability. Consequently, faster convergence speed is possible.

6.3.2 Wilcoxon test

Performance comparison results of the algorithms are collected in Appendix A and are shown in Tables A1–A3 for 10D, in Tables A4–A6 for 30D, in Tables A7–A9 for 50D, and in Tables A10–A12 for 100D.

For each dimension, the comparison of the jSO algorithm which uses LPSR and other algorithms (jSO with other strategy integrated) was conducted using the Wilcoxon signed-rank test with a significance level of $\alpha = 0.05$. Sign '+' means that jSO performs significantly better than the compared algorithm, '-' means that jSO performs significantly worse than the compared algorithm, while '≈' means that there is no significant difference between the compared algorithms. The comparison results are presented as +/≈/- sums at the bottom of each table, and the summarized results are collected in Tables 12 and 13.

For the 10D problems, the baseline jSO with the LPSR strategy consistently outperforms several competing variants of population-size reduction strategies integrated into jSO. In particular, jSO significantly outperforms methods such as jSO+FP and jSO+DSDCDE on a large number of test functions. Overall, the aggregated results show that jSO performs significantly better in 54 cases, compared to only 8 cases where it performs significantly worse, while the majority of comparisons indicate no statistically significant difference. This suggests that, in low-dimensional settings, many population-size adaptation mechanisms integrated into jSO do not yield substantial additional benefits.

Although in the 30D setting, the number of statistically significant wins for jSO increases to 87, the number of losses also rises to 23. jSO with LPSR still outperforms variants such as jSO+FP and jSO+DSDCDE, achieving a large number of significant wins. However, for several algorithms that incorporate more advanced population-size reduction mechanisms (e.g., jSO+NLPSR-ver2, jSO+AGDE-MPP, and jSO+SLDE), the number of jSO losses increases, and a substantial proportion of outcomes remain statistically indistinguishable. This suggests that certain population size adaptation strategies become more competitive in higher-dimensional search spaces, although they do not consistently outperform the LPSR strategy.

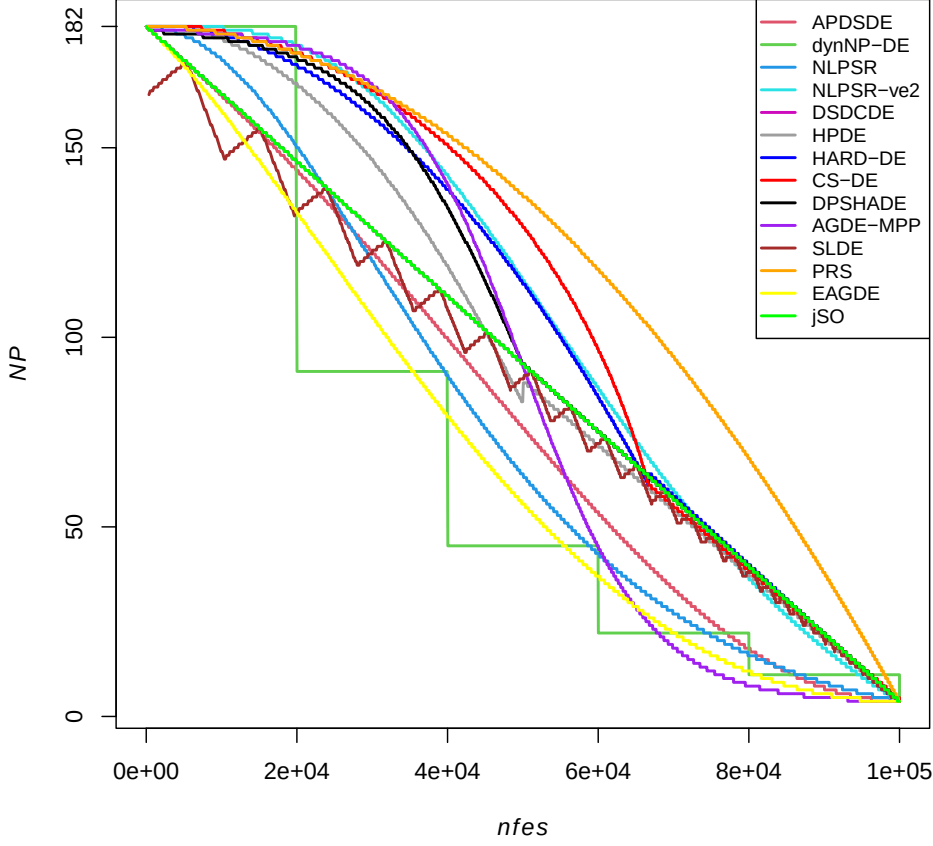


Fig. 2: Reduction/adjustment curve of the population size of the strategies during the evolutionary process for 10D. The initial population size was $NP_{init} = 182$ and final/minimum population size $NP_{end} = 4$.

For the 50D setting, the gap between LPSR's statistically significant wins and losses narrows compared to lower-dimensional settings. This indicates that, as dimensionality increases, several population size adaptation strategies become more competitive. Nevertheless, jSO with LPSR continues to show dominance over certain algorithms, most notably jSO+FP and jSO+DSDCDE, for which jSO with LPSR achieves a large number of significant wins and no losses. In contrast, methods such as jSO+NLPSR-ver2, jSO+HARD-DE, jSO+CS-DE, jSO+DPSHADE, and jSO+AGDE-MPP yield a higher number of statistically significant wins, suggesting that these strategies may provide advantages in more complex search spaces.

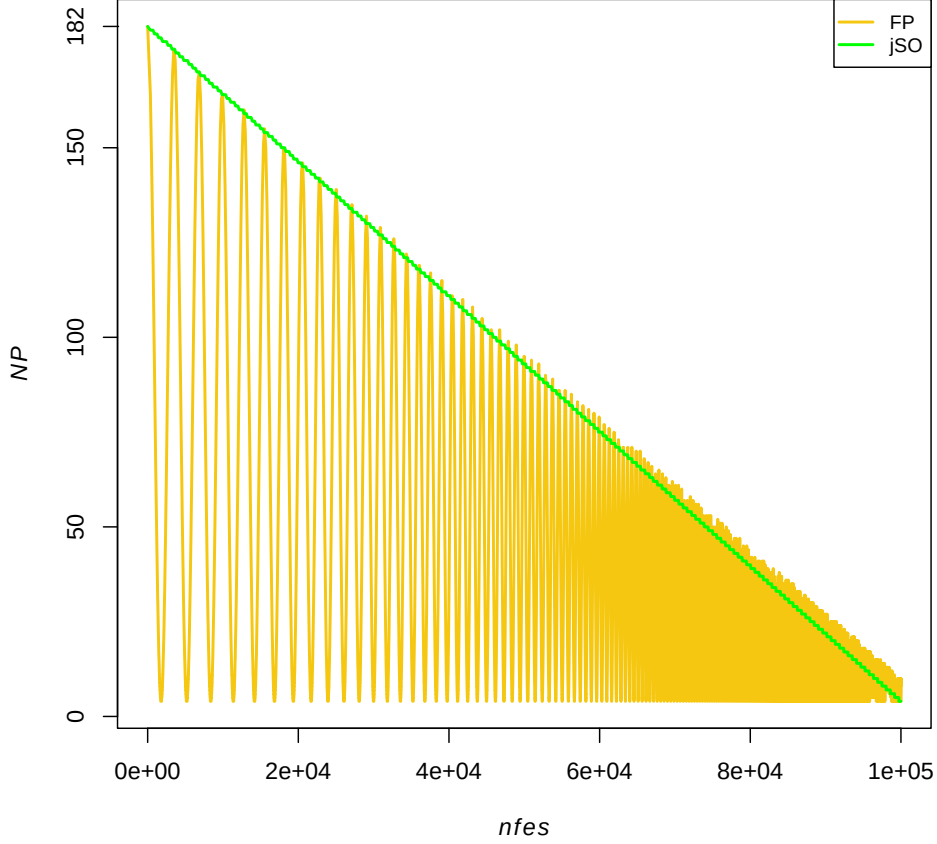


Fig. 3: Reduction/adjustment curve of the population size of the FP strategy and LPSR (jSO) during the evolutionary process for 10D. The initial population size was $NP_{init} = 182$ and minimum population size $NP_{min} = NP_{end} = 4$.

In the 100D setting, performance differences become even more pronounced. This trend suggests that while LPSR remains highly competitive, high-dimensional optimization accentuates the strengths of certain adaptive population size mechanisms. Variants such as jSO+NLPSR-ver2, jSO+AGDE-MPP, jSO+APDSDE, jSO+HARD-DE, jSO+CS-DE, and jSO+SLDE exhibit a larger number of cases where LPSR is significantly outperformed. These observations suggest that non-linear population size reduction or the combination of linear and non-linear reduction strategies may better address the challenges posed by high-dimensional search spaces.

An analysis of the functions that benefit most from population size reduction strategies reveals that, in the 50D setting, the most pronounced improvements are

Table 12: The Wilcoxon signed-rank results of jSO versus other algorithms ($\alpha = 0.05$) on 10D and 30D.

Dimension	Algorithm	$+/\approx/-^1$
10	jSO+APDSDE	1/27/1
	jSO+dynNP-DE	6/22/1
	jSO+NLPSR	0/27/2
	jSO+NLPSR-ver2	0/29/0
	jSO+DSDCDE	14/15/0
	jSO+FP	27/1/1
	jSO+HPDE	0/29/0
	jSO+HARD-DE	0/29/0
	jSO+CS-DE	1/27/1
	jSO+DPSHADE	0/29/0
	jSO+AGDE-MPP	0/29/0
	jSO+SLDE	2/27/0
	jSO+PRS	1/28/0
	jSO+EAGDE	2/25/2
Total		54/344/8
30	jSO+APDSDE	1/26/2
	jSO+dynNP-DE	10/18/1
	jSO+NLPSR	3/24/2
	jSO+NLPSR-ver2	0/26/3
	jSO+DSDCDE	23/6/0
	jSO+FP	29/0/0
	jSO+HPDE	0/28/1
	jSO+HARD-DE	0/28/1
	jSO+CS-DE	0/25/4
	jSO+DPSHADE	0/27/2
	jSO+AGDE-MPP	2/24/3
	jSO+SLDE	6/21/2
	jSO+PRS	6/23/0
	jSO+EAGDE	7/20/2
Total		87/296/23

¹Sign + means that jSO performs significantly better than the compared algorithm, − means that jSO performs significantly worse than the compared algorithm, $\approx$ means that there is no statistical difference between compared algorithms.

observed for functions f4, f6, f7, f10, f14, f19, and f28. Similarly, in the 100D case, per-function performance comparisons indicate that population size adjustment strategies yield significant improvements, particularly for functions f6, f9, f16, and f21.

Across both 50D and 100D settings, the prevalence of statistically indistinguishable results remains high, underscoring the fact that no single strategy consistently dominates across all functions and dimensions. Overall, the results confirm that

Table 13: The Wilcoxon signed-rank results of jSO versus other algorithms ($\alpha = 0.05$) on 50D and 100D.

Dimension	Algorithm	$+/\approx/-^1$
50	jSO+APDSDE	2/23/4
	jSO+dynNP-DE	7/21/1
	jSO+NLPSR	3/23/3
	jSO+NLPSR-ver2	1/21/7
	jSO+DSDCDE	25/4/0
	jSO+FP	29/0/0
	jSO+HPDE	0/27/2
	jSO+HARD-DE	1/22/6
	jSO+CS-DE	1/20/8
	jSO+DPSHADE	1/23/5
	jSO+AGDE-MPP	2/21/6
	jSO+SLDE	4/22/3
	jSO+PRS	5/21/3
	jSO+EAGDE	3/25/1
Total		84/273/49
100	jSO+APDSDE	2/21/6
	jSO+dynNP-DE	16/12/1
	jSO+NLPSR	2/22/5
	jSO+NLPSR-ver2	2/18/9
	jSO+DSDCDE	27/2/0
	jSO+FP	29/0/0
	jSO+HPDE	0/27/2
	jSO+HARD-DE	0/23/6
	jSO+CS-DE	0/23/6
	jSO+DPSHADE	0/27/2
	jSO+AGDE-MPP	2/18/9
	jSO+SLDE	3/20/6
	jSO+PRS	10/15/4
	jSO+EAGDE	6/18/5
Total		99/246/61

¹Sign + means that jSO performs significantly better than the compared algorithm, - means that jSO performs significantly worse than the compared algorithm, $\approx$ means that there is no statistical difference between compared algorithms.

jSO, which uses LPSR, is a strong and robust baseline, while also highlighting that carefully designed population size adaptation strategies can offer advantages in higher-dimensional scenarios. This reinforces the importance of adaptive mechanisms that balance exploration and exploitation as the dimensionality of the optimization problem increases.

Table 14: Algorithms' ranks in 10D.

Algorithm	MeanRank	Rank
jSO+APDSDE	8.1206897	11
jSO+dynNP-DE	9.1379310	13
jSO+NLPSR	6.5344828	4
jSO+NLPSR-ver2	7.7068966	8
jSO+DSDCDE	11.0517241	14
jSO+FP	14.0344828	15
jSO+HPDE	6.9482759	6
jSO+HARD-DE	5.9827586	2
jSO+CS-DE	5.8965517	1
jSO+DPSHADE	7.2586207	7
jSO+AGDE-MPP	7.9137931	9
jSO+SLDE	7.9482759	10
jSO+PRS	6.2931034	3
jSO+EAGDE	8.2758621	12
jSO	6.8965517	5
$\chi^2 = 115.2639, p = 5.2993\text{e-}18$		

Table 15: Algorithms' ranks in 30D.

Algorithm	MeanRank	Rank
jSO+APDSDE	7.2413793	8
jSO+dynNP-DE	10.3620690	13
jSO+NLPSR	7.7068966	9
jSO+NLPSR-ver2	5.9137931	5
jSO+DSDCDE	12.8448276	14
jSO+FP	15.0000000	15
jSO+HPDE	6.7241379	6
jSO+HARD-DE	4.9655172	1
jSO+CS-DE	5.7068966	3
jSO+DPSHADE	5.7413793	4
jSO+AGDE-MPP	5.5517241	2
jSO+SLDE	7.9655172	11
jSO+PRS	7.8620690	10
jSO+EAGDE	9.3620690	12
jSO	7.0517241	7
$\chi^2 = 190.8385, p = 4.0581\text{e-}33$		

6.3.3 Friedman test

Next, we applied the Friedman test with a significance level of $\alpha = 0.05$ and the obtained results for each dimension are presented in Tables 14–17. In each table, Friedman ranks are tabulated, the corresponding performance ordering is provided in the Rank column, and the χ^2 and p values of the Friedman test are reported at the bottom. If the p -value is lower than predefined significance level (α) it indicates that

Table 16: Algorithms' ranks in 50D.

Algorithm	MeanRank	Rank
jSO+APDSDE	7.2931034	8.5
jSO+dynNP-DE	9.7068966	13.0
jSO+NLPSR	7.2586207	7.0
jSO+NLPSR-ver2	5.1206897	2.0
jSO+DSDCDE	13.5344828	14.0
jSO+FP	15.0000000	15.0
jSO+HPDE	6.7068966	6.0
jSO+HARD-DE	4.9137931	1.0
jSO+CS-DE	5.2241379	3.0
jSO+DPSHADE	5.6034483	4.0
jSO+AGDE-MPP	6.2241379	5.0
jSO+SLDE	8.3620690	10.0
jSO+PRS	8.4310345	11.0
jSO+EAGDE	9.3275862	12.0
jSO	7.2931034	8.5
$\chi^2 = 192.2814$, $p = 2.0625\text{e-}33$		

Table 17: Algorithms' ranks in 100D.

Algorithm	MeanRank	Rank
jSO+APDSDE	7.2413793	7
jSO+dynNP-DE	10.3448276	13
jSO+NLPSR	6.5862069	5
jSO+NLPSR-ver2	4.6034483	1
jSO+DSDCDE	13.6206897	14
jSO+FP	15.0000000	15
jSO+HPDE	7.5517241	9
jSO+HARD-DE	6.2241379	4
jSO+CS-DE	5.2586207	2
jSO+DPSHADE	6.8793103	6
jSO+AGDE-MPP	5.4137931	3
jSO+SLDE	7.2758621	8
jSO+PRS	7.7931034	11
jSO+EAGDE	8.6206897	12
jSO	7.5862069	10
$\chi^2 = 174.8261$, $p = 7.238\text{e-}30$		

at least one pair of the compared algorithms have statistical significant difference in performance.

For the 10D problems (Table 14) jSO+CS-DE achieves the best overall rank, closely followed by jSO+HARD-DE and jSO+PRS. The baseline jSO with LPSR attains a competitive mid-level ranking, demonstrating robustness even without more sophisticated population size adaptation mechanisms. In the 30D case (Table 15), a noticeable shift in rankings occurs. jSO+HARD-DE emerges as the top-performing

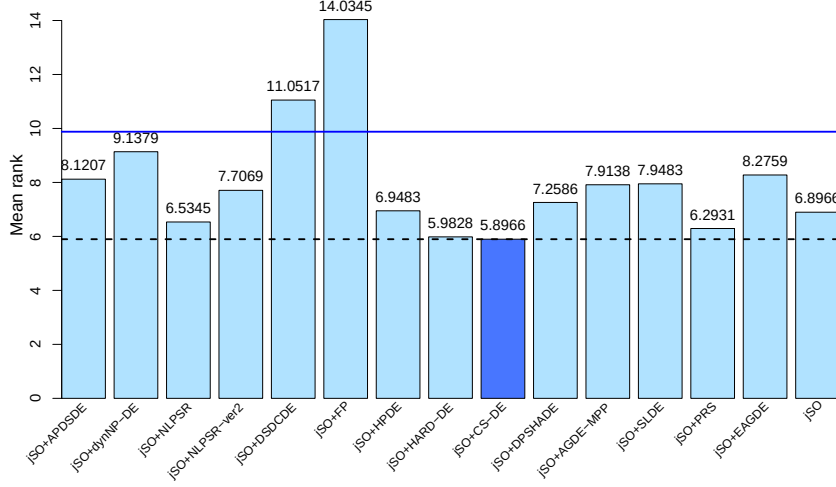


Fig. 4: Friedman ranks of algorithms in 10D with marked critical distance for jSO+CS-DE.

algorithm, followed by jSO+AGDE-MPP and jSO+CS-DE. This change suggests that population size adaptation strategies that are effective in low dimensions may not retain their dominance as dimensionality increases. Nevertheless, the baseline jSO with LPSR remains stable, ranking near the middle of the table, which highlights its consistent performance across varying problem complexities. For 50D problems (Table 16), jSO+HARD-DE continues to dominate, achieving the best rank, with jSO+NLPSR-ver2 and jSO+CS-DE following closely. The baseline jSO and jSO+APDSDE share a similar ranking. At this dimensionality, more sophisticated population size adjustment mechanisms appear increasingly advantageous. In the 100D setting (Table 17), the ranking landscape changes further. jSO+NLPSR-ver2 achieves the best overall rank, followed by jSO+CS-DE and jSO+AGDE-MPP. Overall, the Friedman rankings reveal several important trends. First, while jSO with LPSR provides a strong and reliable baseline across all dimensions, other population size adaptation strategies can offer substantial performance improvements, especially as dimensionality increases. Second, no single strategy dominates across all dimensions, highlighting the dimension-dependent nature of population size control. Finally, strategies that incorporate adaptive, non-linear, or diversity-aware mechanisms tend to perform better in higher-dimensional problems, emphasizing the growing importance of balancing exploration and exploitation as the search space becomes increasingly complex.

The results of the Friedman test are also shown in Figures 4–7. The best-performed algorithm is presented in dark blue. We depict the best-performing algorithm with a dotted line, and the dark blue line shows the critical distance relative to it. The

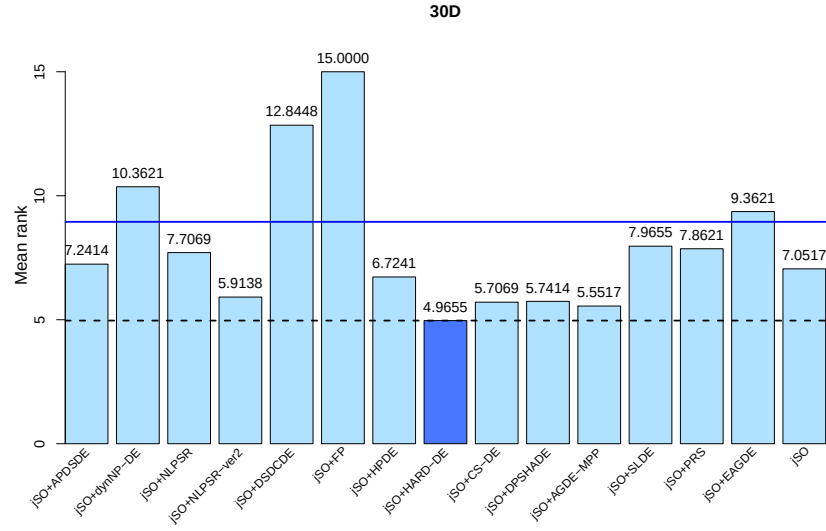


Fig. 5: Friedman ranks of algorithms in 30D with marked critical distance for jSO+HARD-DE.

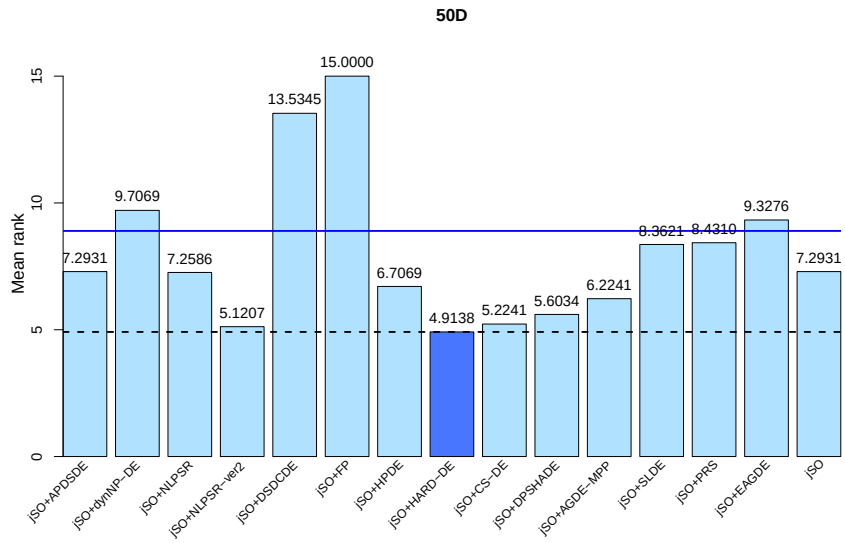


Fig. 6: Friedman ranks of algorithms in 50D with marked critical distance for jSO+HARD-DE.

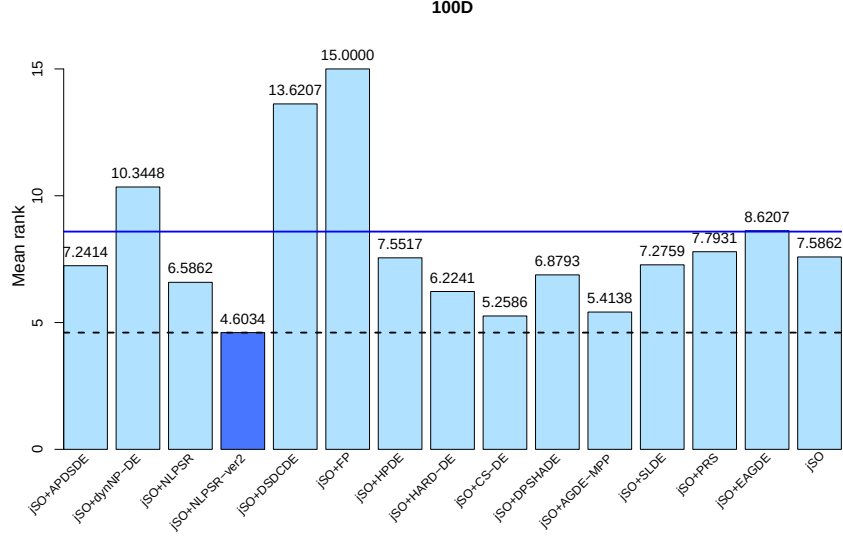


Fig. 7: Friedman ranks of algorithms in 100D with marked critical distance for jSO+NLPSR-ver2.

critical distance indicates that a statistically significant difference exists between two algorithms when the difference in their Friedman rank values exceeds the critical value. For the 10D case, jSO+CS-DE emerges as the best-performing algorithm. With respect to the critical distance (CD) diagram, 12 algorithms lie below the line of statistical significance, while 2 algorithms are positioned above it. For both 30D and 50D, jSO+HARD-DE achieves the best performance. In these dimensions, 10 algorithms fall below the CD curve, whereas 4 algorithms are located above it, indicating statistically significant differences. In the 100D scenario, jSO+NLPSR-ver2 attains the top rank. Notably, this algorithm ranked second in the 50D case. For 100D, 10 algorithms are positioned below the CD curve, one algorithm lies on the curve, and 3 algorithms appear above it. Notably, a considerable performance difference is observed between jSO+NLPSR and jSO+NLPSR-ver2, resulting only from changes in the values of two parameters. Once again, the figures indicate that population size reduction strategies lead to improved performance, with many non-linear reduction methods substantially outperforming linear ones. This advantage becomes increasingly pronounced as the dimensionality of the problem increases.

6.3.4 Convergence graphs

The convergence graphs illustrate how the objective function values evolve throughout the optimization process. The convergence graphs for functions f7 and f15 in the

30D setting are presented in Figure 8, those for functions f19 and f21 in the 50D setting are shown in Figure 9, and the convergence behavior for functions f5 and f15 in the 100D setting is illustrated in Figure 10. Overall, the convergence curves exhibit similar trends across most algorithms, with jSO+FP being a notable exception. Nevertheless, differences in the dispersion of the convergence curves are evident, indicating varying performance levels among the algorithms. These observations suggest that the employed population size regulation strategies significantly influence the convergence behavior throughout the evolutionary optimization process.

Let us present a concise summary of the convergence analysis for six representative cases shown in Figures 8–10. For the two selected 30D functions, the convergence curves of all algorithms remain closely grouped during the first half of the optimization process (up to $MaxFEs/2$) with errors approximately 150 for f7 and 1500 for f15. During the second half of the optimization process, when the population sizes are generally smaller, clearer differences among the convergence curves emerge. The final average error values for f7 across all algorithms, except for two, are between 7 and 10; for f15, they range between 30 and 100, again excluding the same two algorithms. For both functions, no algorithm succeeded in reducing the error to zero. For the two selected 50D functions, the convergence curves of the various algorithms begin to diverge somewhat after the first half of the optimization process (with the error for the better-performing algorithms being half as large). However, significant differences emerge in the convergence curves during the second half of the process; this is reflected in the final average error values, which range between 100 and 200 for function f19 (except for two algorithms) and between 1000 and 2000 for function f21 (except for one algorithm). For both functions, none of the algorithms succeeded in reducing the error to zero. For the first selected 100D function, the convergence curves of all algorithms (with the exception of two) approach the desired value after the first half of the optimization process, as the errors fall below 10^{-3} . However, convergence then stalls. For the second selected 100D function, the convergence curves of all algorithms are closely grouped after the first half of the optimization process (with the standout algorithm jSO+FP achieving the best intermediate result). The final average error values for f15 in the 100D range from 1500 to 4000 across all algorithms.

Overall, these examples show that differences among population-size adjustment strategies tend to become more visible during the later stages of the optimization process. At the same time, the selected cases demonstrate that reaching very high accuracy can remain difficult, particularly for higher-dimensional problems, where convergence may stagnate before the desired error threshold is reached.

Researchers and practitioners often require detailed analyses of DE performance and convergence behavior, particularly in relation to different characteristics of optimization problems. Such in-depth investigations facilitate a deeper understanding of how population size adaptation strategies influence the balance between exploration and exploitation during the evolutionary process. Insights gained from convergence analysis can guide the design of more effective DE variants and support informed decisions when selecting or configuring algorithms for specific problem classes, especially in high-dimensional and complex optimization scenarios.

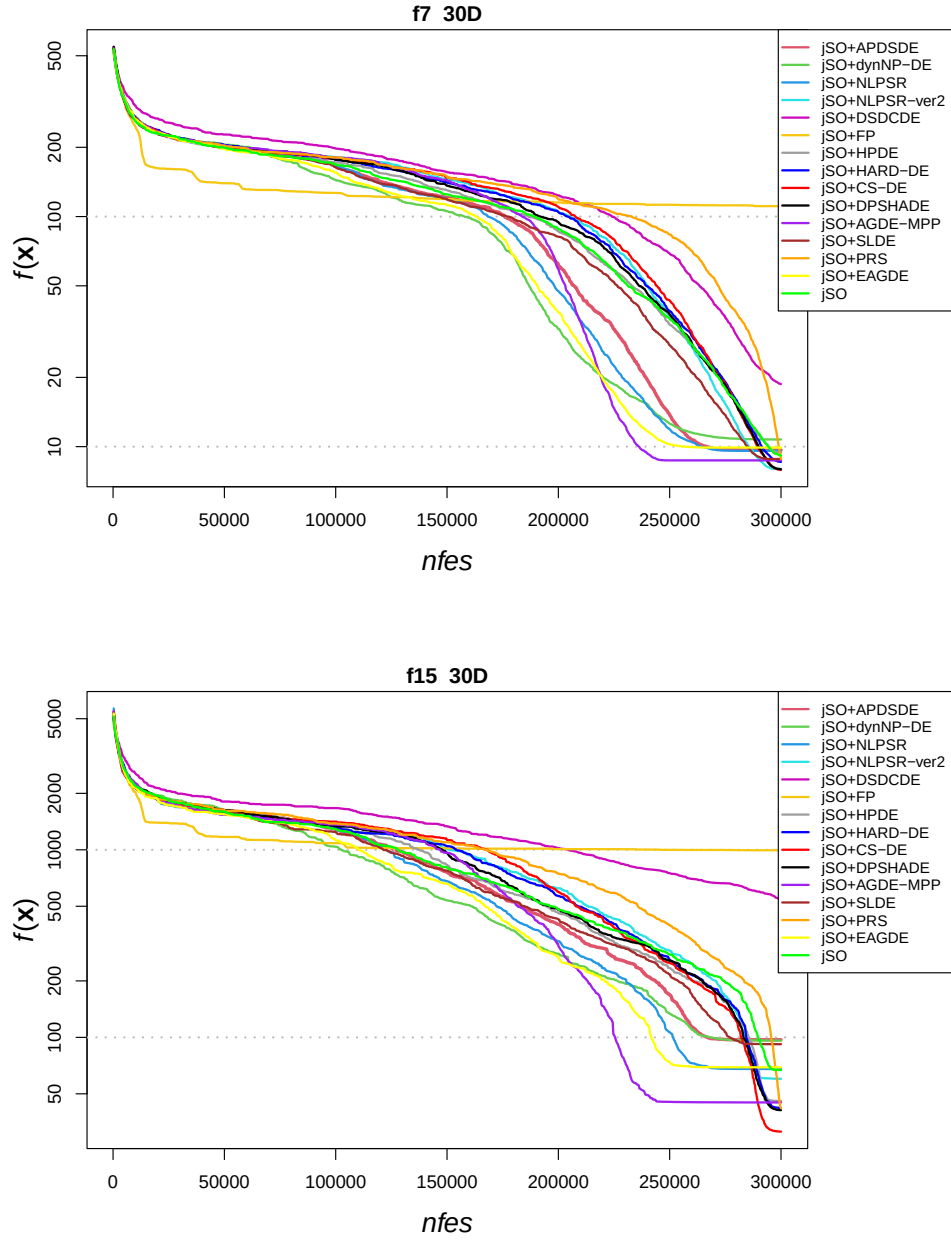


Fig. 8: Convergence graphs of algorithms for functions f7 and f15 in 30D.

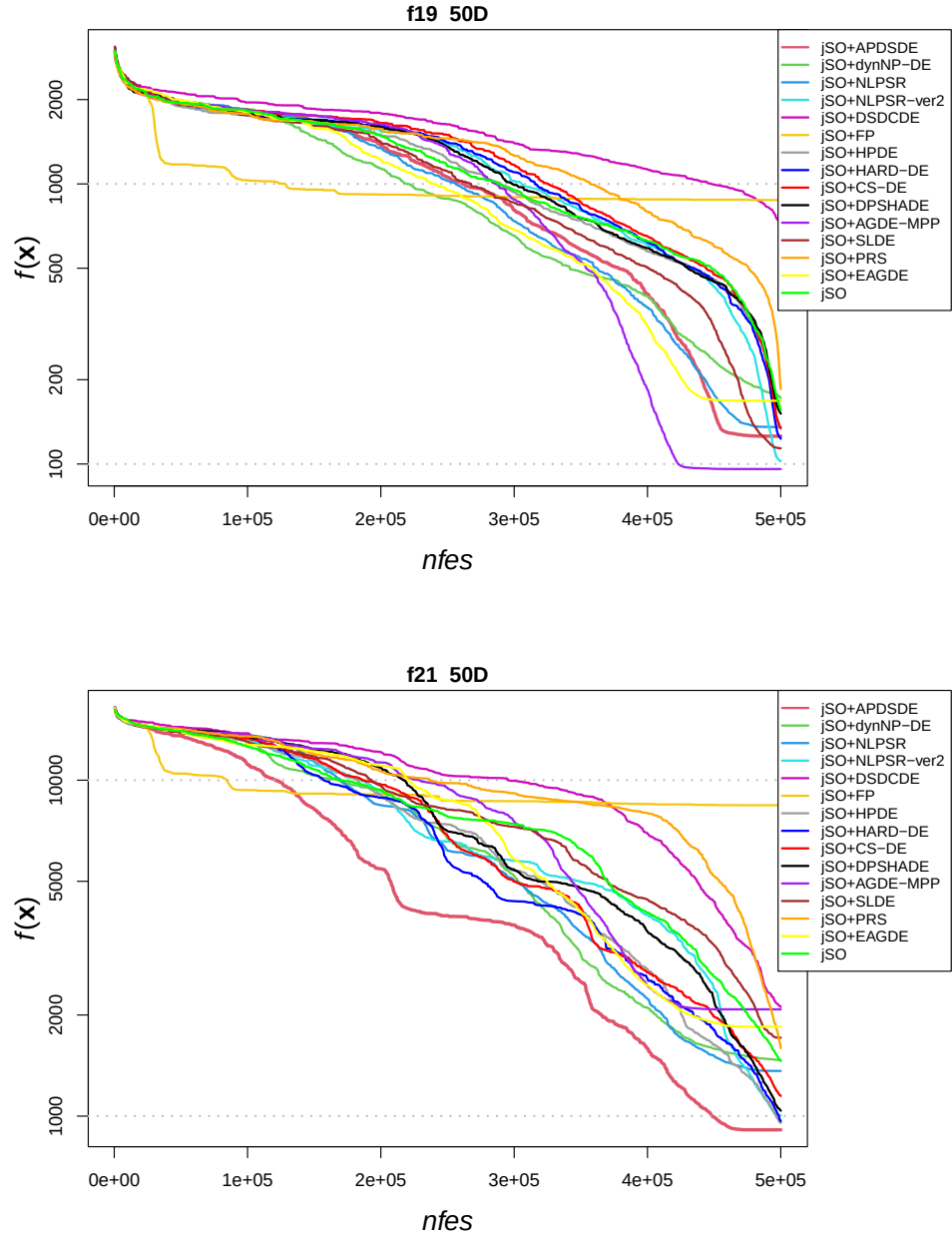


Fig. 9: Convergence graphs of algorithms for functions f19 and f21 in 50D.

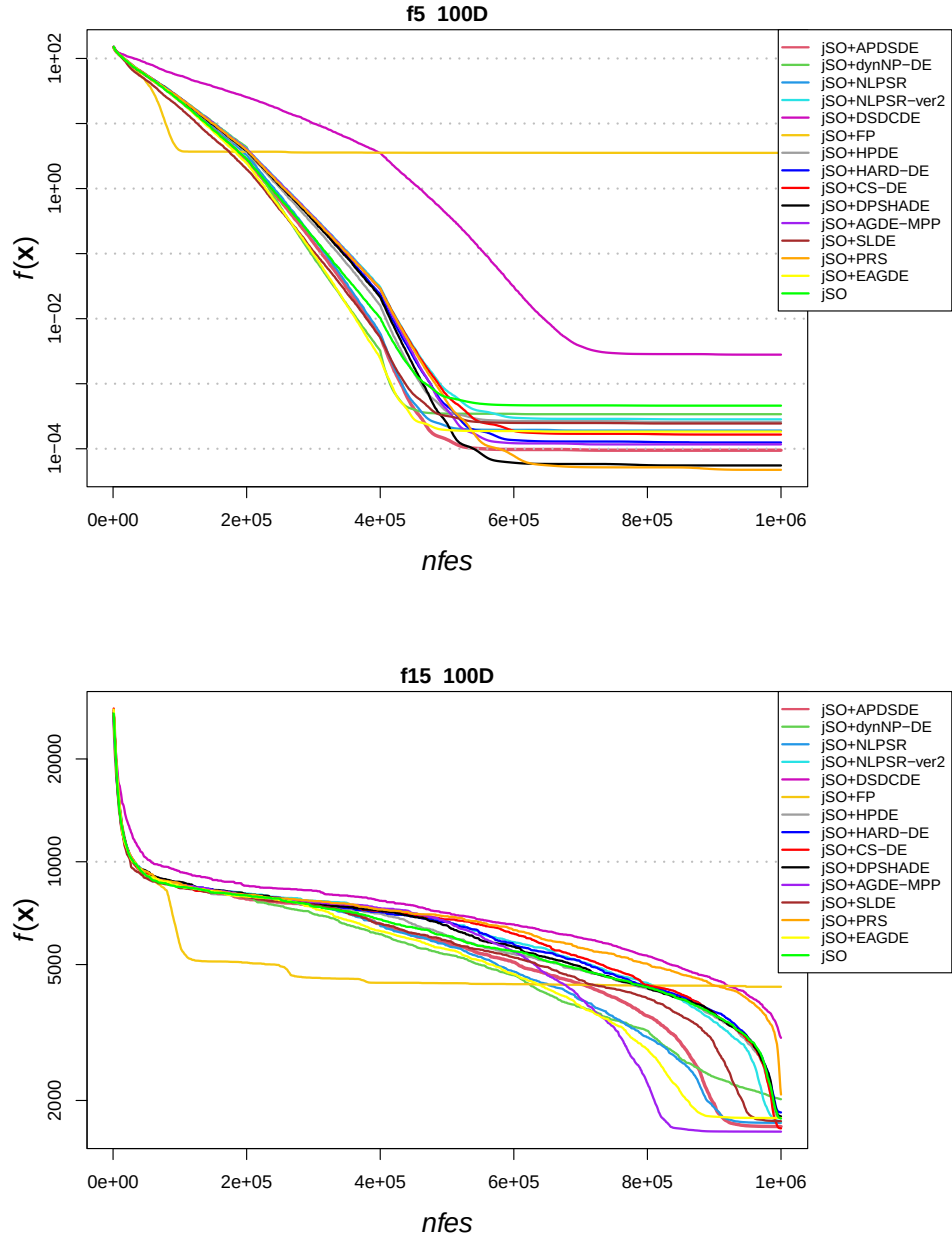


Fig. 10: Convergence graphs of algorithms for functions f5 and f15 in 100D.

6.3.5 Performance analysis by function family and dimension

The figures C1–C16 show Friedman mean ranks of the compared algorithms for different function families and problem dimensions. We can see that, across the four problem dimensions (10D, 30D, 50D, and 100D), the relative performance of the algorithms depends on the function family. This suggests that the effectiveness of a population-size adjustment strategy is related not only to dimensionality, but also to the exploration and exploitation requirements imposed by the problem landscape.

For *unimodal functions*, the differences among most algorithms are relatively small in 10D, 30D, and 50D, indicating that these problems are comparatively easy for all considered methods. Only in 100D do clearer differences emerge, where jSO+dynNP-DE and jSO achieve the best Friedman ranks, but without a significant difference to other algorithms. In contrast, jSO+FP obtains the worst rank across all dimensions. A possible explanation is that unimodal problems contain a single optimum and therefore place less emphasis on preserving population diversity. Consequently, advanced nonlinear or fluctuating population-size strategies offer limited benefit, while frequent or unnecessary changes in population size may even disturb an otherwise straightforward convergence process. The competitive performance of the original jSO with LPSR and dynNP-DE mechanisms is consistent with this interpretation.

For *multimodal functions*, population-size adaptation plays a more important role. In 10D and 50D, jSO+CS-DE achieves the best rank, while jSO+HARD-DE performs best in 30D. For the largest dimension (100D), jSO+AGDE-MPP obtains the best rank. Unlike LPSR, jSO+CS-DE and jSO+HARD-DE do not reduce the population at a constant rate. jSO+CS-DE applies an elliptic reduction in the early stage followed by a linear reduction, whereas jSO+HARD-DE employs a parabolic-linear schedule. Both mechanisms are designed to avoid reducing the population too rapidly during the early part of the search process. Maintaining a larger population for longer may preserve diversity and allow a broader exploration of the search space, which is particularly relevant for multimodal landscapes containing many local optima. The behavior of jSO+AGDE-MPP is even more pronounced in this respect: its hyperbolic-tangent reduction keeps the population relatively large during a substantial part of the evolutionary process and slows the reduction in both the early and late stages. This may help explain its strong performance on 100D multimodal functions, where the number of potentially relevant search regions increases substantially.

For *hybrid functions*, the best-performing method varies with dimensionality. jSO+CS-DE achieves the best rank in 10D and 100D, jSO+DPSHADE performs best in 30D, and jSO+NLPSR-ver2 obtains the best rank in 50D. Hybrid functions combine components with different landscape characteristics and therefore require the optimizer to alternate effectively between broad exploration and more intensive local search. The good performance of jSO+CS-DE and jSO+DPSHADE may be related to their stage-dependent population reduction: both employ different reduction behaviors during different parts of the search rather than following a single linear reduction trajectory. jSO+DPSHADE, for example, applies nonlinear reduction during the first half of the evaluation budget and switches to a linear reduction afterward. Such stage-dependent mechanisms may provide greater flexibility when the landscape contains

regions requiring different search behaviors. The strong performance of jSO+NLPSR-ver2 in 50D also indicates that not only the general form of a nonlinear reduction curve, but also its parameterization can be important. The NLPSR strategy uses two parameters, k_1 and k_2 , to control the population reduction rate during the early and later stages of evolution, respectively. The two NLPSR variants evaluated here differ only in these parameter values, yet their rankings differ across problem families and dimensionalities. This indicates that the timing of population reduction, rather than nonlinear reduction alone, can influence performance.

For *composition functions*, jSO+NLPSR performs best in 10D, whereas jSO+HARD-DE achieves the best Friedman rank in both 30D and 50D. In 100D, jSO+NLPSR-ver2 becomes the top-ranked algorithm. Composition functions combine several basic functions and can therefore contain multiple search regions with substantially different landscape properties. An overly rapid decrease in population size may reduce the algorithm’s ability to explore these regions before sufficient information has been collected. The favorable rankings of jSO+HARD-DE and the two jSO+NLPSR variants suggest that a more gradual or stage-dependent transition from exploration to exploitation can be beneficial on such landscapes. At the same time, the change from jSO+NLPSR in 10D to jSO+NLPSR-ver2 in 100D again suggests that the appropriate reduction rate depends on problem dimensionality.

Considering the results across function families, multimodal, hybrid, and composition functions exhibit substantially larger rank differences than unimodal functions, highlighting the importance of population-size adjustment for difficult optimization problems. Although the original jSO remains competitive, several population-size adjustment mechanisms achieve better rankings across more challenging landscapes, especially for higher dimensional optimization problems. Among them, jSO+CS-DE, jSO+HARD-DE, jSO+NLPSR-ver2, and jSO+AGDE-MPP most frequently achieve the best Friedman ranks, demonstrating the benefits of adaptive population management in maintaining diversity and facilitating the transition from exploration to exploitation. The best-ranked strategies share the property that population reduction is not performed uniformly throughout the search. Instead, they maintain a larger population in the early stages and adjust the reduction rate as the evolutionary process progresses. This may provide a better compromise between preserving population diversity in the early and intermediate stages and concentrating computational effort on promising regions later in the search process.

6.3.6 Computational complexity

In this subsection, we briefly analyze the computational overhead introduced by different population size adjustment strategies. In our experimental work, we used a PC with the following configuration:

System: GNU Linux, CPU: Intel(R) Core(TM) i7-4770 CPU 3.4 GHz, Main memory: 16 GB, Programming language: C++, Compiler: g++.

In technical report for CEC 2024 (Qiao et al. 2023), it is suggested to measure the computing time t_i for the algorithm with 10000 evaluations for function f_i , and

Table 18: Minimum and maximum values of T_2 indicators in seconds.

D	$\min(T_2)$ [s]	$\max(T_2)$ [s]
10	0.33	0.35
30	1.14	1.23
50	2.63	2.73
100	9.32	9.76

compute indicator T_2 as follows:

$$T_2 = \sum_{i=1}^{29} t_i / 29 \quad (104)$$

Recall, there are 29 functions in the CEC 2024 benchmark suite. Since the T_2 indicators obtained for each strategy are quite similar, we only show their minimum and maximum values in Table 18. We can see that the difference between the best and worst-performing strategies among the 15 strategies is negligible across all dimensionalities. For example, the difference is $\approx 5\%$ in the case of 100D.

In fact, the negligible differences among the indicators can be interpreted as follows. The conditions for applying population size adjustment are checked in each iteration, while the strategy itself is triggered only in certain generations. Note that the conditions used and the triggered strategy are also not computationally expensive. Given that, in real-world problems, the total execution time is often dominated by objective function calculations rather than by the computational overhead of the strategy, the additional overhead associated with such a strategy can be considered acceptable.

6.4 Comparison of LPSR against fixed population size

In this subsection, we present the experimental results comparing the jSO algorithm, which employs the LPSR mechanism, with three algorithms using fixed population sizes:

- DE — the classical Differential Evolution algorithm with fixed population size $NP = 100$ (using $F = 0.5$ $CR = 0.9$, and the DE/rand/1 mutation strategy with the binomial crossover),
- jSO-wo-Red — the jSO without population reduction, using a fixed population size $NP = 100$,
- jSO-wo-Half — the jSO without population reduction, using a fixed population size, $NP = NP_{init}/2$. Recall that jSO uses $NP_{init} = 25 \times \log(D)\sqrt{D}$.

The DE and jSO-wo-Red algorithms use the same population size across all problem dimensions, whereas jSO-wo-Half employs a dimension-dependent population size that remains fixed during the whole evolutionary process.

Table 19 shows the summarized results of the Wilcoxon test, while the detailed results are collected in Tables A13–A16. It is evident that the LPSR mechanism

Table 19: The Wilcoxon signed-rank results of jSO versus other algorithms ($\alpha = 0.05$).

Dimension	Algorithm	$+/\approx/-$ ¹
10	jSO-wo-Half	13/16/0
	jSO-wo-Red	13/16/0
	DE	12/13/4
	Total	38/45/4
30	jSO-wo-Half	19/10/0
	jSO-wo-Red	22/7/0
	DE	21/8/0
	Total	62/25/0
50	jSO-wo-Half	21/7/1
	jSO-wo-Red	26/3/0
	DE	25/3/1
	Total	72/13/2
100	jSO-wo-Half	19/9/1
	jSO-wo-Red	24/3/2
	DE	28/1/0
	Total	71/13/3

¹Sign + means that jSO performs significantly better than the compared algorithm, − means that jSO performs significantly worse than the compared algorithm, $\approx$ means that there is no statistical difference between compared algorithms.

generally outperforms the algorithm variants using fixed population sizes across all considered problem dimensions. The advantage becomes more pronounced as the dimensionality increases. For the 10D problems, jSO with LPSR achieves mostly positive results against all compared algorithms, although many comparisons remain statistically indistinguishable. For higher-dimensional problems (30D, 50D, and 100D), the superiority of LPSR becomes more evident. The number of statistically significant wins increases substantially, while the number of losses remains very small.

Figures 11–14 show Friedman ranks of the algorithms with fixed population sizes versus jSO, while the detailed results are collected in Tables A17–A20. Figures confirm the conclusions drawn from Table 19. The jSO algorithm with the LPSR mechanism achieves the best average rank among all compared algorithms across all problem dimensions. In the 10D case, the advantage of LPSR is visible but relatively modest, consistent with the larger number of statistically indistinguishable results observed in the Wilcoxon tests. However, as the dimensionality increases to 30D, 50D, and 100D, the superiority of LPSR becomes substantially more pronounced. The mean rank of jSO with LPSR improves from 1.93 in 10D to approximately 1.2 in higher-dimensional problems, while the competing fixed-population variants remain noticeably worse.

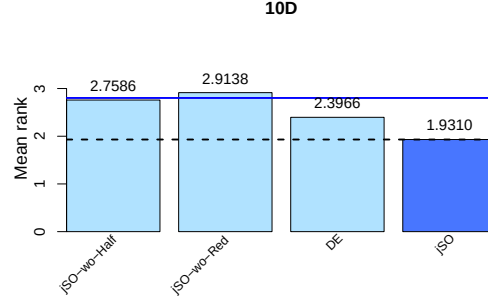


Fig. 11: Friedman ranks of algorithms in 10D with marked critical distance for jSO.

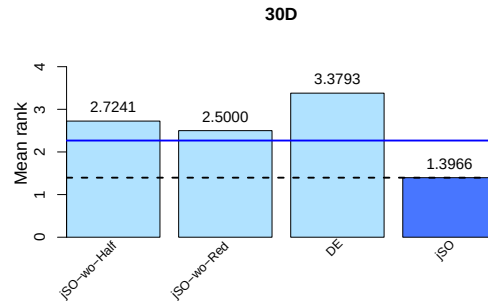


Fig. 12: Friedman ranks of algorithms in 30D with marked critical distance for jSO.

Among the compared algorithms, the classical DE obtains the highest (worst) mean ranks, particularly for 30D–100D problems, indicating that DE with fixed population size is less powerful for complex high-dimensional optimization tasks. The jSO variants without population reduction (jSO-wo-Red and jSO-wo-Half) achieve better results than DE, suggesting that the jSO framework itself contributes positively to performance. Nevertheless, both variants remain clearly inferior to the original jSO with LPSR, highlighting the importance of adaptive population reduction. Overall, the results indicate that the LPSR mechanism improves the search efficiency of jSO, especially as the problem dimensionality increases.

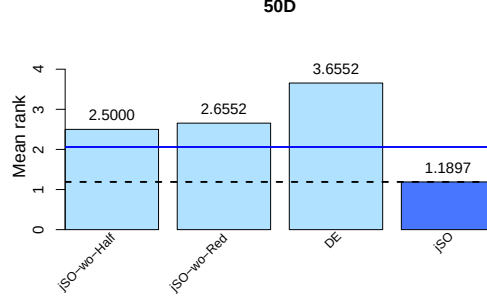


Fig. 13: Friedman ranks of algorithms in 50D with marked critical distance for jSO.

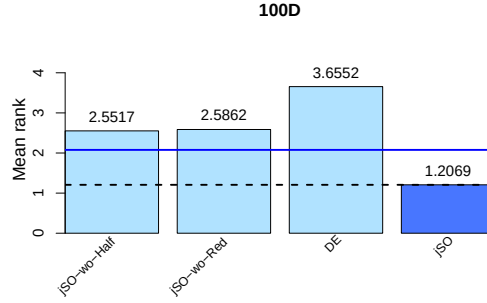


Fig. 14: Friedman ranks of algorithms in 100D with marked critical distance for jSO.

7 Experiment on real-world problems

We evaluate different population-size reduction mechanisms on two real-world problems from the CEC 2011 benchmark suite (Das and Suganthan 2010), originating from two different application domains. The first problem (rwp1) is Parameter Estimation for Frequency-Modulated Sound Waves. The objective is to generate a sound signal that closely matches a given target sound. This problem is highly complex and multimodal, with strong epistatic interactions among variables. The second problem (rwp2) is the Lennard-Jones Potential Problem, which represents a molecular potential energy minimization task. The goal is to minimize the potential energy associated with a pure Lennard-Jones cluster. This is a highly multimodal optimization problem characterized by an exponential number of local optima. The maximum number of fitness function evaluations was set to $MaxFEs = 150,000$ for both real-world problems.

Table 20: The Wilcoxon signed-rank results of jSO versus other algorithms ($\alpha = 0.05$).

Problem	Algorithm	$+/\approx/-^1$
rwp1	jSO+APDSDE	0/1/0
	jSO+dynNP-DE	0/1/0
	jSO+NLPSR	0/1/0
	jSO+NLPSR-ver2	0/1/0
	jSO+DSDCDE	1/0/0
	jSO+FP	1/0/0
	jSO+HPDE	0/1/0
	jSO+HARD-DE	0/1/0
	jSO+CS-DE	0/1/0
	jSO+DPSHADE	0/1/0
	jSO+AGDE-MPP	0/1/0
	jSO+SLDE	0/1/0
	jSO+PRS	0/1/0
	jSO+EAGDE	0/1/0
	Total	2/12/0
rwp2	jSO+APDSDE	0/0/1
	jSO+dynNP-DE	0/0/1
	jSO+NLPSR	0/0/1
	jSO+NLPSR-ver2	0/1/0
	jSO+DSDCDE	1/0/0
	jSO+FP	1/0/0
	jSO+HPDE	0/1/0
	jSO+HARD-DE	0/1/0
	jSO+CS-DE	0/1/0
	jSO+DPSHADE	0/1/0
	jSO+AGDE-MPP	0/0/1
	jSO+SLDE	0/1/0
	jSO+PRS	1/0/0
	jSO+EAGDE	0/0/1
	Total	3/6/5

¹Sign + means that jSO performs significantly better than the compared algorithm, - means that jSO performs significantly worse than the compared algorithm, $\approx$ means that there is no statistical difference between compared algorithms.

Table 20 presents results of the Wilcoxon signed-rank test comparing the original jSO algorithm with different population size reduction strategies. For the first real-world problem rwp1, the results indicate that most algorithms achieve performance statistically similar to jSO, as reflected in 12 cases that are statistically indistinguishable. These results suggest that the considered population reduction mechanisms behave similarly on this problem. For the second real-world problem rwp2, the

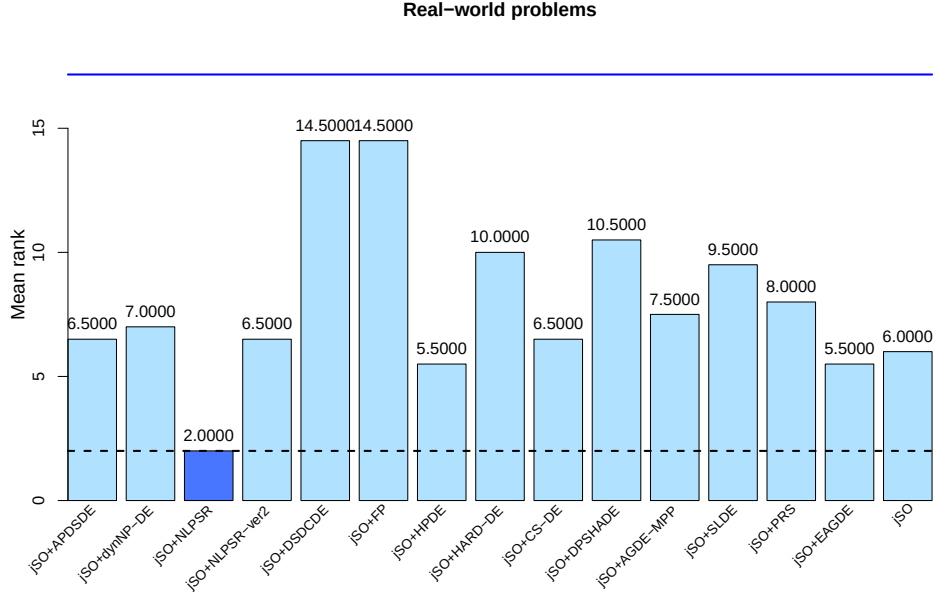


Fig. 15: Friedman ranks of algorithms for real-world problems with marked critical distance for jSO+NLPSR.

results are more diverse. Algorithms jSO+APDSDE, jSO+dynNP-DE, jSO+NLPSR, jSO+AGDE-MPP, and jSO+EAGDE achieve statistically significant improvements over the jSO baseline. Compared to rwp1, the second problem rwp2 appears to be more sensitive to the choice of population size reduction strategy. Readers interested in the detailed results are advised to refer to Tables A21 and A22.

Figure 15 presents the Friedman ranks of the compared population size reduction strategies on the considered real-world problems. The results show that jSO+NLPSR achieves the best overall performance with the lowest mean rank (2.0), indicating that the non-linear population size reduction strategy is the most effective for the considered real-world optimization tasks. Very competitive results are also obtained by jSO+HPDE and jSO+EAGDE, both achieving mean ranks of 5.5. The original jSO algorithm also performs well, obtaining a mean rank of 6.0. On the other hand, the weakest performance is observed for jSO+DSDCDE and jSO+FP, which obtain the highest mean ranks (14.5), suggesting that these population-reduction mechanisms incorporated in jSO are less suitable for the considered real-world problems.

Figure 16 presents convergence graphs for the considered real-world problems. Recall that $MaxFEs = 150,000$ for both problems. It can be observed that the rwp1 problem is solved by almost all algorithms within approximately 100,000 function

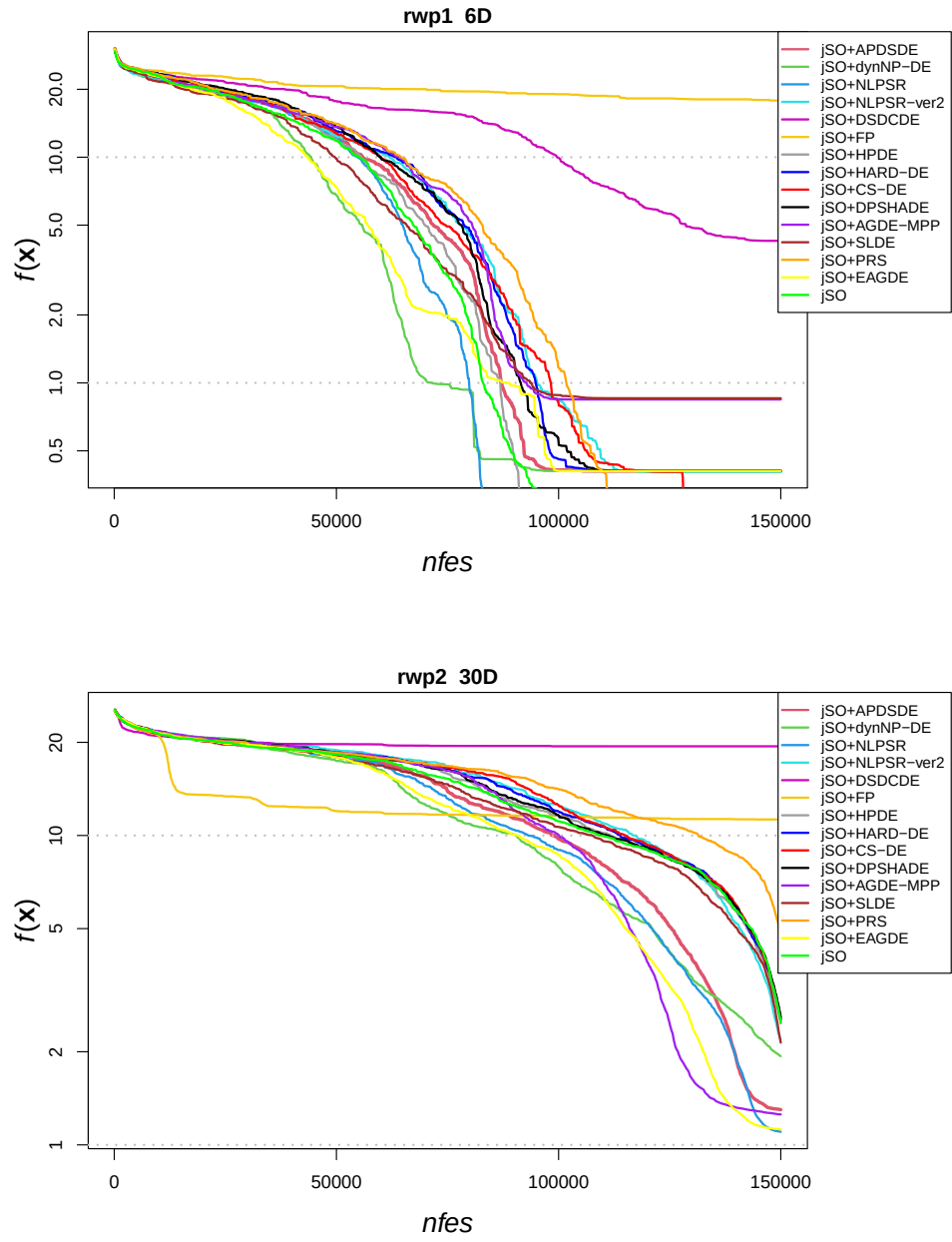


Fig. 16: Convergence graphs of algorithms for real-world problems rwp1 and rwp2.

evaluations, while for obtaining a better solution in the case of the rwp2 problem, a higher number of function evaluations is required.

7.1 An example of convergence behavior

Table 21: Number of CEC 2024 functions (out of 29 for each dimension) for which jSO+NLPSR-ver2 achieved at least one successful run ($SR \geq 4\%$), $SR \geq 50\%$, and $SR = 100\%$.

Dimension	$SR \geq 4\%$	$SR \geq 50\%$	$SR = 100\%$
10D	12	7	6
30D	5	4	3
50D	4	3	3
100D	2	2	1

Before we present the convergence behavior of two population size adjustment algorithms on the real-world problem rwp2, we give a very short overview of the success rate of jSO+NLPSR-ver2, as an example, on the CEC 2024 benchmark test suite. For the jSO-NLPSR-ver2 algorithm, in Table 21 we present the number of cases (out of a total of 29 for each dimension) in which the success rate (SR) is 100%, greater than or equal to 50%, at least 4%. The success rate is defined by the number of successful runs, where a run is successful if the algorithm successfully finds the optimal value of the function, using an error threshold of 10^{-8} . Recall, there are 29 functions in the CEC 2024 benchmark, and 25 runs were performed for each function. In other cases, the function values obtained by the algorithm were far from the optimal values known for this benchmark. The performance strength of the algorithms deteriorated as the dimension increased.

In the next experiment, we increased $MaxFEs$ to 1,500,000 (ten times more) for rwp2 and used jSO and jSO+NLPSR-ver2 in order to analyze convergence behavior. Figure 17 shows the convergence graphs of 25 runs for both algorithms, respectively. SR was 80% (20 of 25) and 96% (24 of 25) for jSO and jSO+NLPSR-ver2, respectively. In these cases, the algorithms converged to the optimum. If we compare the convergence graphs of Figure 17 and the bottom graph in Figure 16, it can be noticed that the success rate increased as the maximal number of function values increased. Furthermore, the convergence speed is lower when $MaxFEs$ is higher (as the algorithm maintains a larger population size for a longer period), whereas a higher success rate is achieved at the end of the optimization process.

7.2 Results of algorithms with fixed population sizes

To further investigate the influence of population size adaptation, we additionally evaluated algorithms using fixed population sizes on the considered real-world problems. The goal of this experiment was to assess whether population reduction mechanisms

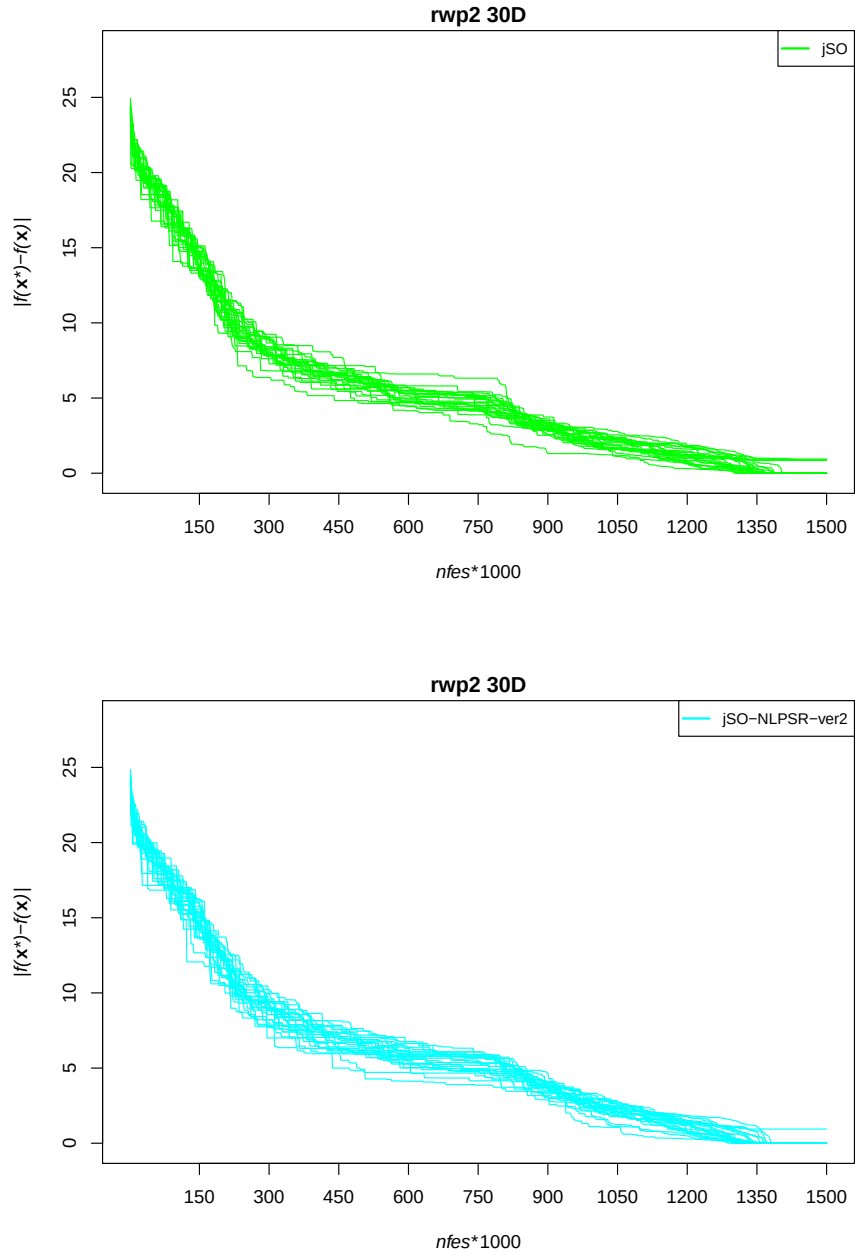


Fig. 17: Convergence graphs of 25 runs of the jSO and jSO+NLPSR-ver2 algorithms for the real-world problem rwp2 with $MaxFEs = 1,500,000$.

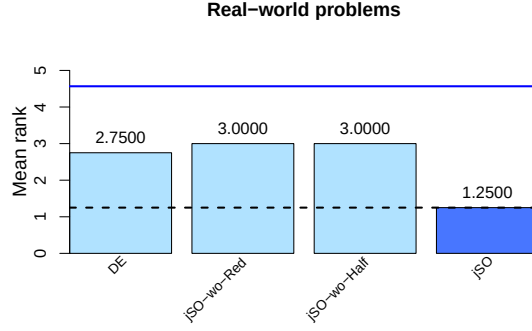


Fig. 18: Friedman ranks of algorithms for real-world problems with marked critical distance for jSO.

provide measurable advantages over fixed population sizes in practical optimization scenarios. The last three rows of Tables A21 and A22 present the results. On rwp1 (Table A21), the classical DE algorithm also reaches the optimum in all runs. In contrast, the jSO without population reduction exhibits noticeably worse performance. The jSO-wo-Red variant achieves a mean error value of 1.66, while jSO-wo-Half obtains a slightly better mean value of 1.257. Both variants also show larger standard deviations, indicating less stable behavior across runs. These results suggest that, for rwp1, the population-reduction mechanism contributes positively to the robustness and reliability of the jSO algorithm. On rwp2 (Table A22), the algorithms using fixed population sizes achieve noticeably worse results compared to most adaptive population reduction strategies.

Figure 18 summarizes Friedman ranks of the algorithms with fixed population sizes on the considered real-world problems. The original jSO algorithm with the LPSR mechanism achieves the best overall performance, obtaining the lowest mean rank (1.25), which clearly indicates its advantage over the fixed-population variants.

8 Discussion, limitations, and recommendations

8.1 Discussion

The experimental results on the CEC 2024 benchmark problems and two real-world problems were obtained using the jSO algorithm, which was extended with various population-size adjustment methods originally integrated into other DE algorithms. Our purpose was not to design new algorithms that would be the most efficient in terms of performance, but only to compare selected methods for population size adjustment during the evolutionary process. The comparison mainly included methods that use NLPSR. However, if the reader is interested in powerful state-of-the-art

algorithms and their comparison, we recommend, among others, the papers (Popič et al. 2025; Brest and Sepesy Maučec 2025; Piotrowska and Piotrowski 2026; Stanovov and Semenkin 2024c). It is worth noting that most of algorithms discussed in the mentioned papers employ LPSR.

In this review, we were mainly interested in methods for adjusting the population size in DE during the evolutionary process. In these methods, bounds are often set on population size variability. Usually, the end user needs to set the population size parameters (NP_{init} and NP_{end}), which determine the size of the population at the beginning and the end of the evolutionary process, respectively. A different possibility can be found in the algorithm of Unbounded DE (UDE) (Kitamura and Fukunaga 2022), which uses a population in which individuals are never replaced and the population grows monotonically as new individuals are produced. UDE uses a parameterized tournament selection.

We integrated the selected population size adjustment strategies into the jSO algorithm. The results indicate which strategies improve performance relative to the original jSO and which do not. As jSO shares key design principles with L-SHADE and several of its derivatives (e.g., iL-SHADE, L-SRTDE, and L-SRTDEwPSM), similar trends may also be observed when these strategies are incorporated into those algorithms. However, this hypothesis would require dedicated experimental verification.

In recent years, as artificial intelligence and machine learning have become dominant across a wide range of scientific and engineering disciplines, their influence has also increasingly extended to evolutionary algorithms. The integration of evolutionary computing with machine learning techniques is a promising research direction for improving optimization algorithms. By learning specific characteristics of a problem, they can achieve more efficient optimization, resulting in a more accurate optimization solution. Learning-based schemes were employed in DE variants to generalize parameter control or to select effective algorithmic configurations (Ye et al. 2025). For example, the Knowledge Learning Evolutionary Computation (KLEC) framework (Jiang et al. 2025) leverages historical successful experiences to construct a knowledge library that guides the evolutionary behavior of individuals. Similarly, a self-learning DE algorithm incorporating a population range indicator was proposed in (Zhao et al. 2023), while Liu et al. (2023) introduced a general-purpose evolutionary algorithm framework grounded in learning mechanisms.

Despite their potential, learning-based approaches introduce several challenges. Designing effective learning strategies that can reliably exploit feedback information during the evolutionary process remains a nontrivial task. Moreover, machine learning-driven methods typically require additional training procedures to derive learning components, which increases computational overhead and often leads to longer runtimes (Ye et al. 2025). These factors must be carefully considered when integrating learning mechanisms into evolutionary algorithms. Learning-based approaches, analogous to other learning mechanisms, could also govern the population-size reduction mechanism.

The literature review in this paper revealed a clear trend toward increasingly sophisticated population-size adjustment mechanisms. Earlier approaches mostly

relied on predefined reduction schedules, whereas more recent algorithms employ heuristic information such as diversity, fitness improvement, or stagnation detection. The experimental study supports this trend, as many of these adaptive mechanisms achieve competitive performance, particularly on higher-dimensional problems.

The experimental results indicate that the advantage of non-linear population size adjustment becomes more pronounced as the dimensionality of the optimization problem increases. A possible explanation is that high-dimensional search spaces require maintaining exploration for a longer portion of the optimization process. Linear population reduction decreases the number of individuals at a constant rate regardless of the search progress, whereas many non-linear strategies delay population reduction or adapt it according to search-state indicators such as diversity, fitness improvement, or stagnation. Consequently, they preserve exploration when it is still beneficial and shift toward exploitation only when the search has already progressed sufficiently.

This interpretation is also consistent with the characteristics of several of the best-performing strategies on the 100-dimensional problems, including those in jSO+NLPSR-ver2, jSO+CS-DE, jSO+AGDE-MPP, and jSO+HARD-DE algorithms. All of these strategies reduce the population size during the evolutionary process and employ either NLPSR mechanism or a combination of LPSR and NLPSR mechanisms. Despite differences in their population size adjustment mechanisms, these strategies generally deviate from the constant-reduction rate of LPSR by maintaining a relatively large population for a longer period during the evolutionary process and applying a more pronounced reduction at another stage of the search process. In high-dimensional search spaces, retaining more individuals for longer may support broader exploration and reduce the risk of prematurely losing potentially useful search directions, while a subsequent faster reduction may allow computational resources to be concentrated on exploitation. The favorable performance of these strategies on the 100-dimensional problems is consistent with this explanation; however, the proposed relationship between their population-size trajectories and performance should be regarded as a hypothesis based on the observed experimental results rather than as a demonstrated causal effect.

Although several non-linear strategies achieved better average performance, no single population-size adjustment mechanism consistently dominated across all benchmark functions and dimensions. This observation is consistent with the No Free Lunch theorem and suggests that the effectiveness of a population-size strategy depends on the characteristics of the optimization landscape, including dimensionality, modality, and the available computational budget.

8.2 Limitations

In this review, we were mainly interested in methods for adjusting the population size in the DE algorithms during the evolutionary process to solve optimization problems. There are several types of optimization: single-objective optimization (SOO), multi-modal optimization (MMO), multi-objective optimization (MOO), large-scale global optimization (LSGO), constrained optimization (CO), etc. This review primarily focuses on SOO and LSGO, but the presented strategies and mechanisms can also

be applied to other types of optimization. Note that algorithms for efficiently solving other types of optimization must also include additional specialized mechanisms.

The next limitation encountered with NLPSR strategies concerns the value of the *MaxFEs* parameter, especially when it is small. The DE algorithms with population size adjustment are more suitable for optimization problems with a larger number of *MaxFEs* and also for problems with larger dimensions. The latter is clearly shown in our real-world problem-solving experiment: classical DE performs well in solving the problem rwp1, which has a small number of dimensions, while the versions with NLPSR and LPRS achieve better performance on rwp2, which has more dimensions than rwp1.

In the presented experimental work, we used the jSO algorithm, which we extended with various population size adjustment methods that were originally integrated into other DE algorithms. Any other algorithm, like L-SHADE, L-SRTDE, etc., could be used instead of jSO. As already mentioned, our intention was not to design new algorithms that would be the most efficient, but rather to compare the selected methods for adjusting the population size during the evolutionary process. Additionally, because NLPSR-based strategies have been frequently used in the literature since 2017, with only a few studies before 2017, we primarily focus on NLPSR methods in our experimental work.

The NP_{init} parameter seems to have a greater impact on the performance of population size adjustment strategies in DE-based algorithms, compared to the NP_{end} parameter. In our experiments, we do not analyze the values of these parameters. Instead, we use $NP_{init} = 25 \times \log(D)\sqrt{D}$, and $NP_{end} = 4$ for all 15 algorithms, as these values are frequently used in the literature.

In practical engineering, scaling the initial population size linearly (e.g., $NP_{init} = 18 \times D$) to counter an exponentially expanding search space often proves computationally prohibitive. This approach can incur excessive costs while yielding diminishing returns in information efficiency. We definitely want to point out this limitation and inform practitioners regarding these scalability bottlenecks. Even before starting the optimization process, it is necessary to obtain the cost, or at least an estimate of it, caused by the evaluation of the criterion function. The cost can also be the execution time. The practitioners' knowledge of the problem being optimized can play an important role in both selecting the optimization algorithm and setting the parameters. In the following section, we present some guidelines.

8.3 Recommendations

Optimization problems with a small number of dimensions can be tackled with simpler algorithms that have fewer parameters, although low dimensionality does not necessarily imply that a problem is easy to solve. More difficult, high-dimensional, and highly multimodal problems may benefit from more advanced optimization algorithms that incorporate multiple mechanisms; however, such algorithms require setting several control parameters. Regardless of whether a simple or an advanced optimization algorithm is used, population size remains an important parameter.

Based on the reviewed studies and experimental results, several practical recommendations for the use of population size adjustment strategies in DE-based

algorithms can be formulated. First, attention should be paid to the selection of the NP_{init} parameter after we know how many iterations ($MaxFEs$) and/or how many generations (G) we have available. Sometimes there are also time limits.

The experimental results provide more specific guidance according to problem characteristics. For unimodal problems, the differences among most population size adjustment strategies were relatively small, particularly at lower dimensions. Therefore, a relatively simple strategy such as LPSR may provide an appropriate starting point, without introducing additional complexity into the population size control mechanism. For multimodal problems, the results favor more flexible non-linear strategies. CS-DE achieved the best Friedman rank in 10D and 50D, HARD-DE in 30D, and AGDE-MPP in 100D. These strategies retain a larger population during selected stages of the evolutionary process, which may help maintain exploration in landscapes with multiple local optima.

For hybrid functions, no single strategy can be recommended across all dimensions. CS-DE obtained the best rank in 10D and 100D, DPSHADE in 30D, and NLPSR-ver2 in 50D. This variability suggests that population size adjustment for hybrid problems should be chosen with particular consideration of dimensionality and the heterogeneous landscape characteristics of these functions. Similarly, for composition functions, NLPSR achieved the best rank in 10D, HARD-DE in 30D and 50D, and NLPSR-ver2 in 100D. Thus, the results generally favor non-linear or stage-dependent reduction strategies for the more complex multimodal, hybrid, and composition function families, particularly at higher dimensions.

These recommendations should not be interpreted as universal rules, since no single population size adjustment strategy consistently dominated across all function families and dimensions, and differences between methods were not always statistically significant. Rather, they provide empirical guidance based on the benchmark problems considered. When little is known about a new optimization problem, LPSR can still serve as a simple reference strategy. For problems known to be highly multimodal or complex, particularly at higher dimensions, the results suggest considering non-linear or stage-dependent strategies such as CS-DE, HARD-DE, AGDE-MPP, or NLPSR-ver2. Therefore, knowledge of the optimization problem and its landscape characteristics remains important when selecting both the optimization algorithm and its population size adjustment strategy.

9 Conclusion

In modern engineering and scientific practice, researchers and practitioners are increasingly required to understand the principles of evolutionary computation and to be proficient in the use of AI-based optimization techniques for solving complex optimization problems. Differential evolution, as a population-based evolutionary algorithm, has proven to be particularly effective for addressing such challenges. While the original DE algorithm employs a fixed population size throughout the evolutionary process, many contemporary variants incorporate advanced strategies that dynamically adjust the population size over time.

This review provides a comprehensive overview of population size reduction methods and presents a comparative analysis of a selected subset of these strategies when integrated into a state-of-the-art DE variant, namely jSO. The comparison focuses primarily on non-linear population size reduction approaches. Extensive experimental evaluations were conducted on the CEC 2024 benchmark suite with problem dimensions of 10, 30, 50, and 100, as well as on two real-world problems, to assess the impact of different population size adjustment strategies during the evolutionary process.

The experimental results demonstrate that several non-linear population size reduction strategies achieved better or highly competitive performance (as indicated by better ranks in the statistical tests) than the linear population size reduction approach employed by the baseline jSO algorithm, particularly for higher-dimensional optimization problems. The statistical analysis identifies different best-performing methods for different problem dimensionalities and function families. Based on Friedman’s rank, jSO+NLPSR-ver2, jSO+CS-DE, and jSO+AGDE-MPP were the top 3 ranked algorithms on D100 using CEC 2024 benchmark functions, and the top 3 algorithms on two real-world problems were jSO+NLPSR, and tied: jSO+HPDE and jSO+EAGDE. All of the aforementioned algorithms belong to the group of NLPSR algorithms. These findings highlight the importance of using adaptive population size control mechanisms and considering non-linear population size adjustment mechanisms as an alternative to the commonly used LPSR. As a note, the parameters of jSO and the selected strategy were kept as recommended in the original paper, since independently tuning them could make the observed difference appear partly attributable to the tuning process. An exhaustive sensitivity analysis of these parameters is a possible direction for future work.

The analysis presented in this review is intended to provide practical guidance for the design of population size adjustment techniques, not only for DE but also for other evolutionary algorithms. By offering insights into how different strategies interact with problem characteristics and dimensionality, this work contributes to the development of more robust and efficient evolutionary optimization methods capable of addressing diverse and challenging optimization landscapes.

A possible direction for further research is how to choose the parameter values for NP_{init} and NP_{end} . Mechanisms aimed at preventing stagnation in the optimization process are used in some strategies (e.g., FADE and DDPSA) to control population size adjustments and remain a promising direction for future research. And finally, the use of learning-based schemes in DE versions for selecting the initial population size (NP_{init}) and for adapting the population size during the evolutionary process would also be interesting for further work.

Acknowledgements

The authors gratefully acknowledge the support of the Slovenian Research and Innovation Agency. Their generous funding and continued support made this work possible.

Statements and Declarations

Author contributions J.B.: Conceptualization, Methodology, Funding, Investigation, Visualization, Supervision, Software, Writing-Original Draft & Editing, and Review. J.P.: Writing-Original Draft, Funding, Analysis, and Review. B.P.: Resources, Writing-Original Draft, and Review. B.B.: Methodology, Supervision, Funding, Writing-Original Draft & Review. M.S.M.: Conceptualization, Methodology, Validation, Funding, Writing & Editing, and Review.

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

Data availability The raw experimental data are publicly available on the Zenodo repository at <https://zenodo.org/records/18621032>. The source code of the experiments can be provided by the corresponding author upon request.

Competing interests The authors have no competing interests to declare that are relevant to the content of this article.

Appendix A Detailed experimental results

Tables A1–A12 present detailed results related to Section 6.3.2, Tables A13 – A20 correspond to Section 6.4, and Tables A21–A22 are related to Section 7.

Table A1: Performance comparison of jSO with other algorithms on 10D. + means that jSO performs significantly better than the compared algorithm.

Func.	jSO+APDSDE				jSO+dynNP-DE				jSO+NLPSR				jSO+NLPSR-ver2				jSO+DSDCDE			
	Mean	Std	sig		Mean	Std	sig		Mean	Std	sig		Mean	Std	sig		Mean	Std	sig	
f1	0	0	≈	0	0	0	≈	0	0	0	≈	0	0	0	≈	0	0	0	≈	0
f2	0	0	≈	0	0	0	≈	0	0	0	≈	0	0	0	≈	0	0	0	≈	0
f3	0	0	≈	0	0	0	≈	0	0	0	≈	0	0	0	≈	0	0	0	≈	0
f4	1.7909	0.7599	≈	2.3083	0.8482	+	≈	1.9899	1.0356	≈	≈	1.7909	0.7035	≈	≈	4.2118	1.5971	≈	≈	4.6760e-07
f5	0	0	≈	0	0	≈	≈	0	0	0	≈	0	0	≈	≈	13.9813	1.5700	+	+	5.0896
f6	12.1232	0.4485	≈	12.2948	0.8127	≈	≈	11.8880	0.6105	≈	≈	11.9824	0.6065	≈	≈	5.0896	2.0471	+	+	0
f7	1.7113	1.0163	≈	2.6665	0.9833	+	≈	2.1491	0.6845	≈	≈	1.7511	0.9209	≈	≈	336.5353	184.0606	+	+	1.0373
f8	0	0	≈	0	0	≈	≈	0	0	0	≈	0	0	≈	≈	1.0373	1.2199	+	+	44.6284
f9	69.9566	94.2237	≈	79.5863	80.5447	≈	≈	42.4325	67.1283	≈	≈	33.7202	67.6432	≈	≈	4.6284	83.7425	+	+	3.0317
f10	0	0	≈	0	0	≈	≈	0	0	0	≈	0	0	≈	≈	3.8710	5.9426	+	+	0.2240
f11	0.3747	0.1803	≈	10.3507	33.0582	≈	≈	0.3247	0.1708	≈	≈	0.3663	0.1506	≈	≈	1.1284	0.4835	+	+	5.3233
f12	3.5547	2.3410	≈	3.1101	2.3708	≈	≈	2.5505	2.3712	≈	≈	2.5477	2.4142	≈	≈	0.2916	0.2062	≈	≈	0.1503
f13	9.9347e-02	0.2870	≈	0.2610	0.4773	+	≈	5.1873e-02	0.2055	≈	≈	0.1194	0.3300	≈	≈	0.8693	1.4526	≈	≈	112.5460
f14	0.3378	0.2007	≈	0.3095	0.1924	≈	≈	0.3404	0.1934	≈	≈	0.2788	0.2256	≈	≈	304.2385	2.3112	+	+	236.4421
f15	0.5139	0.2798	≈	0.5842	0.3606	≈	≈	0.6080	0.3722	≈	≈	0.6085	0.2552	≈	≈	403.4019	15.0675	≈	≈	0
f16	0.2110	0.2806	-	0.3977	0.3874	≈	≈	0.3873	0.3846	≈	≈	0.5176	0.4817	≈	≈	389.4565	0.1700	≈	≈	389.4565
f17	0.2582	0.2134	≈	0.2614	0.2124	≈	≈	0.2496	0.2208	≈	≈	0.2582	0.2184	≈	≈	335.1720	97.3230	≈	≈	335.1720
f18	1.6877e-02	2.2912e-02	≈	2.4935e-02	2.5286e-02	+	≈	1.2484e-02	1.2459e-02	≈	≈	1.3634e-02	1.1265e-02	≈	≈	252.4705	8.2266	+	+	252.4705
f19	0.3371	8.6437e-02	≈	1.2316	3.3950	≈	≈	0.4019	0.2216	≈	≈	0.3746	0.1274	≈	≈	398.1056	10.2183	+	+	398.1056
f20	170.2534	49.1923	+	145.7433	52.6753	≈	≈	128.8811	47.2720	≈	≈	120.6652	42.1833	≈	≈	7.9937e-02	7.9937e-02	≈	≈	7.9937e-02
f21	100.0000	0	≈	100.0000	0	≈	≈	100.0000	0	≈	≈	100.0000	0	≈	≈	100.0160	7.9937e-02	≈	≈	100.0160
f22	301.5607	1.7354	≈	302.2574	1.7068	+	≈	301.3969	1.6640	≈	≈	300.9191	1.3701	≈	≈	304.2385	2.3112	+	+	304.2385
f23	310.6883	63.4650	≈	256.2860	109.4307	≈	≈	264.3831	104.6511	≈	≈	274.6924	100.2047	≈	≈	236.4421	114.2464	≈	≈	236.4421
f24	412.4844	21.5996	≈	407.0155	18.5423	≈	≈	403.3454	15.0776	-	≈	405.2127	16.9874	≈	≈	403.4019	15.0675	≈	≈	403.4019
f25	300.0000	0	≈	300.0000	0	≈	≈	300.0000	0	≈	≈	300.0000	0	≈	≈	300.0000	0	≈	≈	300.0000
f26	389.3540	0.2440	≈	389.3950	0.2234	≈	≈	389.3540	0.2440	≈	≈	389.4565	0.1700	≈	≈	389.4565	0.1489	≈	≈	389.4565
f27	311.3498	56.7490	≈	324.9457	86.3395	≈	≈	322.6987	78.5624	≈	≈	335.1720	97.3230	≈	≈	334.0490	94.1064	≈	≈	334.0490
f28	235.4171	3.8167	≈	233.3390	2.6248	-	≈	232.5711	2.7757	-	≈	234.3001	2.7972	≈	≈	252.4705	8.2266	+	+	252.4705
f29	398.8136	21.3753	≈	395.0640	2.5993	+	≈	394.5213	4.7178e-02	≈	≈	3.3082e+04	1.6344e+05	≈	≈	398.1056	10.2183	+	+	398.1056
+ / ≈ / -	1/27/1				6/23/1				0/27/2				0/29/0				14/15/0			

Table A2: Performance comparison of jSO with other algorithms on 10D. + means that jSO performs significantly better than the compared algorithm.

Func.	jSO+FP			jSO+HPDE			jSO+HARD-DE			jSO+CS-DE			jSO+DPSHADE		
	Mean	Std	sig	Mean	Std	sig	Mean	Std	sig	Mean	Std	sig	Mean	Std	sig
f1	1.8587e+07	1.9437e+07	+	0	0	≈	0	0	≈	0	0	≈	0	0	≈
f2	773.2287	734.5505	+	0	0	≈	0	0	≈	0	0	≈	0	0	≈
f3	8.6336	0.8200	+	0	0	≈	0	0	≈	0	0	≈	0	0	≈
f4	30.0468	7.4659	+	1.7511	0.6600	≈	1.7511	0.7198	≈	1.4725	0.7105	≈	1.8705	0.7221	≈
f5	7.6649	3.1513	+	0	0	≈	0	0	≈	0	0	≈	0	0	≈
f6	59.5678	9.5271	+	11.9031	0.6060	≈	11.7379	0.5995	≈	12.0876	0.6461	≈	12.0240	0.6184	≈
f7	29.8480	8.5153	+	1.5919	0.9526	≈	1.9103	0.6988	≈	1.5919	0.9083	≈	1.9512	0.8844	≈
f8	44.6794	25.2646	+	0	0	≈	0	0	≈	0	0	≈	0	0	≈
f9	1246.1571	190.0306	+	31.7249	57.2253	≈	24.9774	44.3574	≈	29.2117	56.3067	≈	39.8348	64.0352	≈
f10	36.4454	17.9753	+	0	0	≈	0	0	≈	0	0	≈	0	0	≈
f11	1.3997e+06	1.4591e+06	+	0.3663	0.1506	≈	0.3580	0.1648	≈	0.2914	0.1590	≈	0.2997	0.1908	≈
f12	5280.9695	2734.5411	+	3.6636	2.2543	≈	3.5423	2.2495	≈	3.0347	2.3893	≈	3.5343	2.3432	≈
f13	71.4882	19.3187	+	3.9798e-02	0.1990	≈	0	0	≈	3.9798e-02	0.1990	≈	0.1194	0.3300	≈
f14	230.2170	111.1842	+	0.2989	0.2096	≈	0.2379	0.2097	≈	0.2250	0.2292	≈	0.2756	0.2282	≈
f15	82.9402	48.4499	+	0.4991	0.2458	≈	0.5799	0.2927	≈	0.5301	0.2967	≈	0.5720	0.2544	≈
f16	57.7451	10.6814	+	0.3781	0.3720	≈	0.3660	0.3150	≈	0.5740	0.3769	≈	0.5517	0.4233	≈
f17	8921.4207	4558.1563	+	0.2694	0.2216	≈	0.2882	0.2139	≈	0.2033	0.1860	≈	0.2216	0.2120	≈
f18	140.4802	149.9769	+	8.5858e-03	1.1388e-02	≈	9.3631e-03	1.1441e-02	≈	1.7148e-02	1.3022e-02	+	9.3511e-03	1.1429e-02	≈
f19	72.7457	19.3538	+	0.3621	0.1168	≈	0.3871	0.1361	≈	0.3621	0.1168	≈	0.4019	0.2216	≈
f20	110.4120	5.8044	≈	145.3799	52.2577	≈	137.0845	50.4703	≈	141.1816	51.4832	≈	137.0714	50.4487	≈
f21	115.6712	4.9648	+	100.0115	5.7275e-02	≈	96.0000	20.0000	≈	100.0000	0	≈	100.0116	5.8033e-02	≈
f22	308.6341	58.1280	+	301.2874	1.6514	≈	301.0388	1.5677	≈	301.2793	1.4786	≈	301.5653	1.7527	≈
f23	179.0743	66.5473	-	274.0404	99.8234	≈	237.2658	114.3989	≈	300.4635	75.7965	≈	292.2567	85.6655	≈
f24	430.1678	20.6642	+	407.0525	18.5229	≈	408.8738	19.7725	≈	401.5506	12.5753	-	401.5739	12.5695	≈
f25	377.8205	28.2102	+	300.0000	0	≈	300.0000	0	≈	300.0000	0	≈	300.0000	0	≈
f26	395.6717	2.2477	+	389.4360	0.1917	≈	389.4155	0.2092	≈	389.4360	0.1917	≈	389.3950	0.2234	≈
f27	443.0755	92.6719	+	335.1716	97.3218	≈	335.1720	97.3230	≈	322.6987	78.5624	≈	323.8222	82.5506	≈
f28	305.1821	17.0675	+	234.6262	3.2341	≈	234.2256	2.7355	≈	234.8354	2.6846	≈	234.0935	2.6016	≈
f29	3.6165e+05	4.1662e+05	+	394.5207	4.7043e-02	≈	394.5090	9.8025e-03	≈	394.5196	4.4278e-02	≈	394.5265	5.7325e-02	≈
+ / ≈ / -	27 / 1 / 1			0 / 29 / 0			0 / 29 / 0			1 / 27 / 1			0 / 29 / 0		

Table A3: Performance comparison of jSO with other algorithms on 10D. + means that jSO performs significantly better than the compared algorithm.

Func.	jSO+AGDE-MPP				jSO+SLDE				jSO+PRS				jSO+EAGDE				jSO	
	Mean	Std	sig		Mean	Std	sig		Mean	Std	sig		Mean	Std	sig		Mean	Std
f1	0	0	0	≈	0	0	0	≈	0	0	0	≈	0	0	0	≈	0	0
f2	0	0	0	≈	0	0	0	≈	0	0	0	≈	0	0	0	≈	0	0
f3	0	0	0	≈	0	0	0	≈	0	0	0	≈	0	0	0	≈	0	0
f4	1.8705	0.7771	0	≈	2.1491	0.9386	+	≈	1.4733	0.7669	0	≈	2.0695	1.1094	0	≈	1.5521	0.8164
f5	0	0	0	≈	0	0	0	≈	0	0	0	≈	0	0	0	≈	0	0
f6	12.1302	0.6403	0	≈	12.1236	0.5145	≈	≈	11.9537	0.6330	0	≈	12.1051	0.6424	0	≈	11.9474	0.5542
f7	2.0297	0.9298	0	≈	1.8705	0.8285	≈	≈	2.0300	0.5358	0	≈	1.8705	0.7221	0	≈	1.9899	0.8124
f8	0	0	0	≈	0	0	0	≈	0	0	0	≈	0	0	0	≈	0	0
f9	43.6272	78.4526	0	≈	44.9960	81.3814	≈	≈	32.2424	57.4446	0	≈	55.6398	67.3822	0	≈	32.4858	46.7644
f10	0	0	0	≈	0	0	0	≈	0	0	0	≈	0	0	0	≈	0	0
f11	5.1475	23.9604	0	≈	5.0976	23.9707	≈	≈	0.3081	0.1486	0	≈	5.5944	23.9686	0	≈	0.3580	0.1412
f12	3.2302	2.2412	0	≈	2.5210	2.3955	≈	≈	2.2972	2.3306	0	≈	3.2194	2.5379	0	≈	3.2180	2.3938
f13	0.1116	0.3102	0	≈	7.9597e-02	0.2755	≈	≈	7.9598e-02	0.2755	0	≈	0.1434	0.4617	0	≈	3.9798e-02	0.1990
f14	0.3289	0.2194	0	≈	0.2621	0.2310	≈	≈	0.2355	0.2155	0	≈	0.3254	0.2108	0	≈	0.2653	0.2056
f15	0.6081	0.2920	0	≈	0.6309	0.3022	≈	≈	0.6003	0.2920	0	≈	0.6086	0.4551	0	≈	0.5638	0.2759
f16	0.4441	0.4401	0	≈	0.4580	0.3613	≈	≈	0.7532	0.5340	+	≈	1.0708	4.0315	0	≈	0.4262	0.2664
f17	0.2806	0.1856	0	≈	0.2307	0.2234	≈	≈	0.2537	0.2110	0	≈	0.2751	0.2111	0	≈	0.2865	0.2186
f18	1.3430e-02	1.4559e-02	0	≈	5.4409e-03	8.9048e-03	≈	≈	7.9831e-03	1.1497e-02	0	≈	1.4050e-02	1.3260e-02	0	≈	9.3511e-03	1.2751e-02
f19	0.4394	0.2298	0	≈	0.4620	0.1592	≈	≈	0.3746	0.1274	0	≈	0.3122	9.0117e-02	0	≈	0.3871	0.1632
f20	133.2168	49.4218	0	≈	149.4204	52.5118	≈	≈	124.5700	44.6286	0	≈	145.2855	52.1447	0	≈	145.2757	59.5918
f21	100.0000	0	0	≈	100.0000	0	0	≈	100.0000	0	0	≈	100.0000	0	0	≈	100.0000	0
f22	301.2539	1.6193	0	≈	302.0072	1.9041	+	≈	300.5274	1.0780	0	≈	301.7193	1.7239	0	≈	300.6974	1.4499
f23	256.2044	109.3731	0	≈	283.6258	93.7387	≈	≈	263.5924	104.2863	0	≈	256.1663	109.3404	0	≈	264.9539	104.9940
f24	399.7438	9.0822	0	≈	412.4569	21.6335	≈	≈	407.0148	18.5385	0	≈	405.2006	16.9900	0	≈	408.8673	19.7697
f25	300.0000	0	0	≈	300.0000	0	0	≈	300.0000	0	0	≈	300.0000	0	0	≈	300.0000	0
f26	389.4360	0.1917	0	≈	389.4360	0.1917	≈	≈	389.4360	0.1917	0	≈	389.4360	0.1917	0	≈	389.4360	0.1917
f27	312.4729	62.3644	0	≈	312.4729	62.3644	≈	≈	369.2197	125.8262	0	≈	300.0000	0	0	≈	324.9457	86.3395
f28	234.0901	3.2855	0	≈	233.7653	3.2505	≈	≈	235.1403	2.4024	0	≈	233.6533	3.7576	0	≈	235.1833	3.7476
f29	394.5343	6.1088e-02	0	≈	394.5281	5.2502e-02	≈	≈	394.5118	3.2559e-02	0	≈	394.5358	6.2895e-02	0	≈	394.5191	4.6988e-02
+ / ≈ / -	0 / 29 / 0				2 / 27 / 0				1 / 28 / 0				2 / 25 / 2					

Table A4: Performance comparison of jSO with other algorithms on 30D. + means that jSO performs significantly better than the compared algorithm.

Func.	jSO+APDSDE			jSO+dynNP-DE			jSO+NLPDR			jSO+NLPDR-ver2			jSO+DSDCDE		
	Mean	Std	sig	Mean	Std	sig	Mean	Std	sig	Mean	Std	sig	Mean	Std	sig
f1	0	0	≈	0	0	≈	0	0	≈	0	0	≈	0	0	≈
f2	0	0	≈	0	0	≈	0	0	≈	0	0	≈	0	0	≈
f3	58.5616	0	≈	58.5616	0	≈	58.5616	0	≈	58.5616	0	≈	58.5616	0	≈
f4	7.8801	2.0089	≈	9.1153	1.4009	≈	9.0342	2.0293	≈	7.8033	1.6919	≈	18.6204	4.2572	+
f5	4.8400e-08	1.7552e-07	≈	3.1104e-06	6.3218e-06	+	1.7520e-07	4.3770e-07	+	2.9200e-08	7.4606e-08	≈	5.0240e-06	7.9585e-06	+
f6	39.3648	1.8346	≈	39.8765	1.5984	≈	38.9370	2.0217	≈	37.6715	1.6254	≈	46.8296	4.8754	+
f7	9.6710	1.6875	≈	10.7458	1.9269	+	9.5914	1.9242	≈	7.9999	1.4500	≈	18.7256	4.0536	+
f8	0	0	≈	0	0	≈	0	0	≈	0	0	≈	0	0	≈
f9	1464.5898	290.3240	≈	1646.8172	191.0362	+	1448.5812	225.6466	≈	1490.9712	267.0641	≈	2852.6389	752.1999	+
f10	2.9822	2.3164	≈	3.8632	3.1920	≈	2.7305	2.3840	≈	2.4736	1.8194	≈	15.1615	19.0227	+
f11	178.1358	114.1047	≈	280.7713	162.3070	≈	208.2313	135.5537	≈	163.7629	126.8935	≈	958.7735	376.3686	+
f12	14.3134	4.9457	≈	14.9213	4.7287	≈	13.0743	5.4147	≈	15.6974	4.1751	≈	26.5671	11.8147	+
f13	21.9688	0.9498	≈	21.8847	3.8751	+	22.2557	1.3453	≈	21.2537	1.0115	≈	28.3133	5.0762	+
f14	1.3310	0.7098	≈	1.6768	0.9700	≈	1.1197	0.7854	≈	1.0671	0.6578	≈	15.3999	18.1058	+
f15	97.0803	95.3367	≈	96.1326	82.0665	≈	67.7904	69.9137	≈	60.1338	68.1294	≈	542.7153	156.1725	+
f16	31.0924	10.5243	≈	37.8432	8.2049	≈	30.6807	8.7906	≈	34.2514	5.0917	≈	98.6176	42.8742	+
f17	20.7374	0.4043	≈	21.3432	0.7526	≈	20.9272	0.6249	≈	20.6660	0.2009	≈	34.4229	11.9971	+
f18	5.1807	2.0530	≈	5.6133	1.6400	+	5.8083	1.8798	+	4.9057	1.8727	≈	13.0081	3.0256	+
f19	26.2538	6.9677	≈	32.8137	7.2336	+	28.5021	6.2268	≈	27.2863	5.1222	≈	174.7211	67.6817	+
f20	209.6307	1.7017	≈	211.7503	1.8834	+	210.2212	1.6888	+	207.7731	2.2107	≈	220.2932	3.6127	+
f21	100.0000	0	≈	100.0000	0	≈	100.0000	0	≈	100.0000	0	≈	100.0000	0	≈
f22	352.3785	3.5803	+	352.1973	3.5088	+	350.3439	3.2783	≈	350.1983	3.8065	≈	368.4063	6.0258	+
f23	426.9665	2.0130	≈	427.2826	2.2685	+	426.0231	2.3499	≈	424.8083	1.7920	≈	439.7301	6.0087	+
f24	386.6971	9.8674e-03	≈	386.7032	1.0197e-02	+	386.6970	8.0397e-03	≈	386.6989	8.3684e-03	≈	386.7207	1.9876e-02	+
f25	945.9467	43.4341	≈	959.6154	35.1931	≈	935.1898	36.7701	≈	926.8934	37.1096	≈	1129.2843	66.2775	+
f26	493.9697	8.5310	≈	497.9608	5.8187	≈	496.6564	6.3772	≈	496.9504	6.5941	≈	505.8193	4.2665	+
f27	309.1180	31.5583	≈	304.1314	20.6571	≈	312.8219	35.4825	≈	321.9399	44.8483	≈	313.2494	36.6628	≈
f28	426.1379	19.8559	≈	433.7798	10.9759	≈	430.4980	7.3472	≈	436.2038	13.3030	≈	501.0567	27.4711	+
f29	1969.9324	11.3156	≈	1971.3169	23.5011	≈	1972.3677	25.2055	≈	1965.1135	8.8459	≈	2037.9077	74.2556	+
+ / ≈ / -	1/26/2			10/18/1			3/24/2			0/26/3			23/6/0		

Table A5: Performance comparison of jSO with other algorithms on 30D. + means that jSO performs significantly better than the compared algorithm.

Func.	jSO+FP			jSO+HPDE			jSO+HARD-DE			jSO+CS-DE			jSO+DPSHADE		
	Mean	Std	sig	Mean	Std	sig	Mean	Std	sig	Mean	Std	sig	Mean	Std	sig
f1	2.7108e+07	1.4820e+07	+	0	0	≈	0	0	≈	0	0	≈	0	0	≈
f2	7991.2001	3501.1909	+	0	0	≈	0	0	≈	0	0	≈	0	0	≈
f3	108.9340	6.6612	+	58.5616	0	≈	58.5616	0	≈	58.5616	0	≈	58.5616	0	≈
f4	105.9363	16.2017	+	8.6062	1.8158	≈	8.1480	1.9600	≈	7.5187	1.5857	≈	8.8038	2.1503	≈
f5	10.0509	1.9137	+	1.1200e-08	3.8764e-08	≈	0	2.8390e-08	≈	0	0	≈	6.0400e-08	2.1944e-07	≈
f6	169.2557	26.0914	+	38.6351	1.6203	≈	37.7868	1.3266	≈	37.8678	1.7843	≈	37.6328	1.8764	≈
f7	110.9834	26.2355	+	9.4133	2.2366	≈	8.5834	2.1507	≈	7.9305	1.5079	≈	7.9759	1.6648	≈
f8	283.9977	135.0349	+	0	0	≈	0	0	≈	0	0	≈	0	0	≈
f9	6192.4076	610.2559	+	1506.1486	275.7604	≈	1421.3702	232.6984	≈	1579.3453	267.8310	≈	1549.7890	267.9465	≈
f10	170.1792	25.9488	+	4.4757	11.9939	≈	2.6425	1.6019	≈	4.5018	11.9450	≈	2.6851	2.2931	≈
f11	4.0316e+06	3.3326e+06	+	188.3123	153.8097	≈	154.5877	67.5293	≈	158.4050	123.4892	≈	145.1480	102.2415	≈
f12	1.4737e+05	9.9352e+04	+	13.7008	6.3915	≈	15.8082	4.7931	≈	17.6354	2.2866	≈	15.7927	5.5654	≈
f13	339.2798	131.2449	+	21.1780	4.3306	≈	21.3652	0.9465	≈	21.3379	1.2242	≈	22.1337	1.4322	≈
f14	2.8849e+04	1.4888e+04	+	1.2471	0.6993	≈	1.2219	0.8494	≈	0.6729	0.2585	≈	0.9444	0.6887	≈
f15	994.4511	341.9198	+	45.7778	56.0809	≈	41.8662	45.9190	≈	31.4273	50.1633	≈	40.9066	45.4334	≈
f16	236.1487	83.3962	+	32.3957	8.0713	≈	32.4253	6.4180	≈	33.1798	6.2508	≈	31.3350	8.1102	≈
f17	3.9171e+04	1.8674e+04	+	20.7732	0.3169	≈	20.7589	0.2753	≈	20.7714	0.2899	≈	20.7113	0.2699	≈
f18	2.1597e+04	1.8336e+04	+	4.6013	2.2662	≈	4.0616	1.5312	≈	4.0992	1.0023	≈	3.7547	1.1210	≈
f19	328.9823	98.9380	+	27.8821	5.4926	≈	29.5596	4.0993	≈	29.5435	6.0255	≈	28.3477	8.3033	≈
f20	304.6681	19.6655	+	209.3032	1.6273	≈	208.1680	1.3720	≈	207.4741	2.0203	≈	208.7631	2.2132	≈
f21	156.4072	21.1918	+	100.0000	0	≈	100.0000	0	≈	100.0000	0	≈	100.0000	0	≈
f22	473.4874	21.8028	+	351.0008	3.2390	≈	349.9695	3.3341	≈	349.2014	3.0270	≈	349.7954	3.0022	≈
f23	529.4105	24.2234	+	426.7418	2.3472	≈	425.6248	2.4273	≈	425.9987	2.6222	≈	425.4266	2.1917	≈
f24	404.2013	12.5133	+	386.6968	7.2204e-03	≈	386.6967	7.3689e-03	≈	386.6980	7.4916e-03	≈	386.6967	7.0365e-03	≈
f25	2063.9527	192.9630	+	934.5819	37.5188	≈	923.7332	30.0626	≈	942.0643	41.9780	≈	931.4129	37.5325	≈
f26	522.7379	6.5793	+	495.2519	7.4186	≈	495.6130	6.8268	≈	497.1922	9.5780	≈	496.0999	7.2238	≈
f27	474.0864	22.0672	+	308.6904	30.1179	≈	300.0000	0	≈	300.0000	0	≈	309.1180	31.5583	≈
f28	764.4716	114.8427	+	435.1500	15.3625	≈	429.4596	22.9587	≈	430.6040	14.1077	≈	430.5964	23.0100	≈
f29	1.3762e+05	9.6564e+04	+	1967.2592	7.6916	≈	1968.4458	10.0960	≈	1967.1176	5.9715	≈	1973.4097	21.6588	≈
+ / ≈ / -	29/0/0			0/28/1			0/28/1			0/25/4			0/27/2		

Table A6: Performance comparison of jSO with other algorithms on 30D. + means that jSO performs significantly better than the compared algorithm.

Func.	jSO+AGDE-MPP				jSO+SLDE				jSO+PRS				jSO+EAGDE				jSO			
	Mean	Std	sig		Mean	Std	sig		Mean	Std	sig		Mean	Std	sig		Mean	Std	sig	
f1	0	0	≈		0	0	≈		0	0	≈		0	0	≈		0	0	≈	
f2	0	0	≈		0	0	≈		0	0	≈		0	0	≈		0	0	≈	
f3	58.5616	0	≈		58.5616	0	≈		58.5616	0	≈		58.5616	0	≈		58.5616	0	≈	
f4	8.1587	1.9691	≈		9.2332	2.2344	≈		8.9933	2.0671	≈		9.1138	2.4486	≈		8.5503	1.9831	≈	
f5	6.8000e-08	1.0025e-07	+		0	0	≈		1.6800e-08	4.6433e-08	≈		1.9040e-07	6.3442e-07	+		1.1200e-08	3.8764e-08	≈	
f6	39.0540	1.9555	≈		39.4201	2.2548	≈		40.6769	2.1080	+		39.1526	1.9170	≈		38.9737	1.6484	≈	
f7	8.7158	1.9327	≈		8.8352	2.3035	≈		8.8248	2.9390	≈		9.9098	2.1004	≈		9.1448	1.8277	≈	
f8	0	0	≈		0	0	≈		0	0	≈		0	0	≈		0	0	≈	
f9	1480.9740	230.3699	≈		1468.4424	280.1149	≈		1740.6653	261.1413	+		1614.6004	201.5795	+		1464.8674	248.8887	≈	
f10	2.6558	2.0113	≈		2.9189	1.9076	≈		2.2705	1.9886	≈		5.4012	11.8176	≈		2.8312	2.0799	≈	
f11	138.0158	133.6496	≈		169.9254	109.2630	≈		153.1057	121.0918	≈		247.8282	137.3728	≈		216.3273	167.1318	≈	
f12	14.9876	3.6295	-		15.4783	4.6031	≈		18.5520	5.2352	≈		14.6231	4.8867	-		17.0228	5.2018	≈	
f13	21.6121	1.2103	≈		22.8482	1.3566	+		22.1094	1.0283	≈		22.0905	1.0117	+		20.0149	5.4341	≈	
f14	1.1447	0.5495	≈		1.2149	0.7798	≈		1.6177	0.9340	≈		1.2158	0.7313	≈		1.3631	0.9776	≈	
f15	44.9698	66.1696	-		91.9732	118.0278	≈		41.8526	64.4093	≈		69.2152	90.7205	≈		66.9240	72.1669	≈	
f16	34.0291	7.1161	≈		28.3796	8.8317	-		41.9529	5.7653	+		34.4238	7.8047	≈		36.3462	6.8523	≈	
f17	20.6201	0.3575	-		20.5893	0.4068	-		20.8227	0.2735	≈		20.6596	0.3329	-		20.9854	0.5952	≈	
f18	5.8014	1.3433	+		4.2143	1.3799	≈		4.9476	1.9220	≈		6.3863	2.0030	+		4.5346	1.5415	≈	
f19	29.5236	7.0840	≈		27.0207	8.2952	≈		33.9566	6.5601	+		30.7029	9.1382	≈		29.3567	5.6791	≈	
f20	208.1694	1.1903	≈		211.0922	3.1487	+		209.1704	1.9164	≈		210.6937	1.9045	+		208.5554	2.3110	≈	
f21	100.0000	0	≈		100.0000	0	≈		100.0000	0	≈		100.0000	0	≈		100.0000	0	≈	
f22	349.4312	2.7310	≈		353.8092	4.6763	+		351.2636	2.8797	≈		352.3965	3.3496	+		349.6782	2.9115	≈	
f23	426.0860	2.8112	≈		430.9163	3.4203	+		426.2791	1.9338	≈		427.6096	2.9799	+		425.8976	1.7728	≈	
f24	386.6964	7.7768e-03	≈		386.6970	6.2616e-03	≈		386.6974	7.4854e-03	≈		386.6985	6.9327e-03	≈		386.6972	8.4031e-03	≈	
f25	928.8209	56.2552	≈		967.8600	37.8845	+		926.4856	37.0351	≈		931.5949	33.1733	≈		944.7799	45.1769	≈	
f26	495.4257	5.1173	≈		498.4148	4.8752	+		494.1093	7.5998	≈		495.8577	7.8707	≈		493.9173	7.8760	≈	
f27	300.0000	0	≈		309.1180	31.5583	≈		300.0000	0	≈		304.5590	22.7950	≈		312.8219	35.4825	≈	
f28	425.1577	20.8675	≈		431.2400	14.4688	≈		448.8406	18.9009	+		431.5408	7.6709	≈		425.6575	27.0707	≈	
f29	1970.1912	7.5975	≈		1972.3262	26.5924	≈		1968.2553	10.3375	≈		1975.0742	27.4009	≈		1971.9225	11.0916	≈	
+ / ≈ / -	2/24/3				6/21/2				6/23/0				7/20/2							

Table A7: Performance comparison of jSO with other algorithms on 50D. + means that jSO performs significantly better than the compared algorithm.

Func.	jSO+APDSDE				jSO+dynNP-DE				jSO+NLPSPR				jSO+NLPSPR-ver2				jSO+DSDCDE			
	Mean	Std	sig		Mean	Std	sig		Mean	Std	sig		Mean	Std	sig		Mean	Std	sig	
f1	0	0	≈		0	0	≈		0	0	≈		0	0	≈		7.6560e-07	3.4043e-06	+	
f2	0	0	≈		0	0	≈		0	0	≈		0	0	≈		2.5075e-04	5.1329e-04	+	
f3	47.0697	38.4488	≈		40.4939	34.2325	≈		68.8036	51.0080	+		43.9471	41.1317	≈		51.3541	44.5382	≈	
f4	15.6408	2.9082	≈		17.9015	3.4473	≈		16.5163	2.0105	≈		13.7197	3.3328	−		36.2972	9.3386	+	
f5	6.1000e-07	9.0507e-07	≈		1.6145e-05	9.8208e-06	+		3.1996e-06	4.3657e-06	+		3.4840e-07	5.3769e-07	+		6.5662e-05	7.5949e-05	+	
f6	65.2938	2.3312	−		68.0342	2.9175	≈		65.5191	2.5941	≈		64.1848	3.1377	−		82.2427	7.7544	+	
f7	16.9939	3.1973	≈		18.0774	2.8093	≈		16.3571	2.9161	≈		15.5159	3.5155	≈		39.0940	7.8935	+	
f8	0	0	≈		0	0	≈		0	0	≈		0	0	≈		2.1754e-02	9.1877e-02	≈	
f9	2875.0930	426.6198	−		3384.3979	340.9098	≈		3011.2304	364.9782	≈		3164.3122	355.7654	≈		4766.8449	608.4006	+	
f10	28.2022	3.3570	≈		26.1315	3.4904	−		26.0474	4.1343	−		27.0823	3.7138	≈		37.8638	7.0309	+	
f11	1801.0762	454.9568	≈		1851.1234	410.3644	≈		1885.6697	312.9729	≈		1686.1516	406.7212	≈		1.0397e+04	7810.4325	+	
f12	24.4299	15.5866	≈		34.6527	26.3330	≈		29.5880	16.9679	≈		29.4835	17.9093	≈		119.8791	46.8410	+	
f13	25.6237	1.8144	≈		26.7666	2.3872	≈		25.4980	2.5791	≈		24.6285	2.4143	≈		38.4617	5.8434	+	
f14	23.6068	3.1548	≈		26.4119	3.5825	≈		23.7584	2.8751	≈		22.0574	1.8578	−		73.8675	28.9639	+	
f15	435.9682	190.8774	≈		438.1788	161.2538	≈		433.2880	133.1984	≈		407.1610	132.6577	≈		966.6311	175.4810	+	
f16	295.9383	103.6980	≈		281.5061	126.9607	≈		286.7175	84.9622	≈		253.6034	93.8675	≈		929.5026	149.6284	+	
f17	24.9250	1.9854	≈		27.1688	2.8918	+		24.5467	2.3383	≈		23.2130	1.5910	≈		93.4895	43.5738	+	
f18	15.9777	1.9924	+		16.6203	2.2005	≈		15.5496	2.1320	+		13.0239	3.0400	≈		37.9743	12.5811	+	
f19	125.6859	71.1683	−		172.1733	99.4664	≈		135.4631	90.3963	−		102.3795	46.3925	−		705.1396	218.7004	+	
f20	217.3568	2.1802	≈		219.8491	3.3105	+		218.7463	2.5650	≈		215.3956	3.2062	−		238.3763	9.7485	+	
f21	910.5069	1478.0584	≈		1465.5995	1868.1425	≈		1363.1407	1724.5559	≈		1003.9259	1498.4506	≈		2115.3462	2757.0525	≈	
f22	433.0608	6.0600	≈		430.7096	5.8214	≈		430.2081	6.5707	≈		429.0872	5.9898	−		465.9973	10.2496	+	
f23	507.6991	3.6233	≈		509.1915	3.7351	+		506.7652	3.3882	≈		507.2014	3.6988	≈		540.0895	8.2027	+	
f24	480.7152	2.3332	≈		480.1350	0.5776	≈		480.7221	2.3188	≈		481.9851	4.4289	≈		483.1144	5.0850	+	
f25	1145.7269	58.2157	≈		1154.9810	41.3961	≈		1141.0843	38.5170	≈		1147.4784	46.8971	≈		1476.3469	108.7666	+	
f26	513.3643	9.7989	≈		514.8695	9.7482	+		511.4204	11.1330	≈		514.3402	10.0861	+		547.0828	22.8119	+	
f27	460.8026	9.7692	≈		458.8487	0	≈		458.8487	0	≈		458.8487	0	≈		458.8487	0	≈	
f28	338.7798	11.6542	−		366.8860	15.6680	≈		345.2721	15.3290	−		358.2482	11.7058	≈		518.4432	108.3486	+	
f29	6.1225e+05	2.7163e+04	≈		6.0858e+05	4.0848e+04	≈		6.0517e+05	2.5435e+04	≈		6.1415e+05	3.6585e+04	≈		6.3831e+05	5.5908e+04	+	
+ / ≈ / −	2/23/4				7/21/1				3/23/3				1/21/7				25/4/0			

Table A8: Performance comparison of jSO with other algorithms on 50D. + means that jSO performs significantly better than the compared algorithm.

Func.	jSO+FP				jSO+HPDE				jSO+HARD-DE				jSO+CS-DE				jSO+DPSHADE			
	Mean	Std	sig		Mean	Std	sig		Mean	Std	sig		Mean	Std	sig		Mean	Std	sig	
f1	3.5968e+07	1.1175e+07	+	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
f2	2.3114e+04	8039.3543	+	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
f3	225.0967	21.1751	+	48.8863	39.9466	0	0	0	56.2337	44.0616	0	0	46.6174	40.4422	0	0	40.8476	33.0963	0	0
f4	211.7930	29.8832	+	15.2985	3.2748	0	0	0	14.8433	2.6448	0	0	13.8747	3.1473	0	0	14.7049	2.1988	0	0
f5	8.1979	1.5549	+	5.6960e-07	1.1776e-06	0	0	0	2.4200e-07	3.2652e-07	0	0	1.8120e-07	4.6488e-07	0	0	4.3560e-07	4.9760e-07	0	0
f6	285.4338	35.3347	+	65.2230	2.6181	0	0	0	64.7329	2.1173	0	0	63.4985	2.2365	0	0	65.7340	3.4842	0	0
f7	217.3290	26.4542	+	16.0731	3.6147	0	0	0	14.6586	3.7064	0	0	13.5796	2.2949	0	0	15.1186	3.1542	0	0
f8	311.3026	83.5256	+	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
f9	1.0754e+04	798.4998	+	3243.0253	345.8680	0	0	0	3347.5126	354.8276	0	0	3257.9153	284.0756	0	0	3159.6663	357.2724	0	0
f10	267.8982	50.8139	+	28.5980	3.4211	0	0	0	25.7241	3.4107	0	0	24.9732	3.5338	0	0	26.6902	3.7240	0	0
f11	5.8600e+06	1.9628e+06	+	1706.5955	375.1121	0	0	0	1652.4092	321.1851	0	0	1746.3436	385.7257	0	0	1676.1011	405.2186	0	0
f12	1.4541e+05	1.0646e+05	+	28.1597	18.4816	0	0	0	25.6606	16.6372	0	0	30.9120	15.9920	0	0	31.5458	16.3810	0	0
f13	657.1492	236.9511	+	24.4086	1.7712	0	0	0	24.4126	2.0290	0	0	24.1394	2.1891	0	0	25.2443	2.2498	0	0
f14	2.8132e+04	1.3621e+04	+	22.4134	2.3310	0	0	0	22.4321	2.1624	0	0	22.2217	2.0833	0	0	22.5007	1.7953	0	0
f15	1640.8502	367.3136	+	439.7848	130.4253	0	0	0	418.4034	134.9286	0	0	381.8778	152.3600	0	0	408.6304	163.0237	0	0
f16	1140.9256	196.2743	+	318.8309	120.5996	0	0	0	252.6807	106.0692	0	0	278.2626	96.3001	0	0	277.0087	114.6814	0	0
f17	3.5303e+04	1.3810e+04	+	25.0727	2.1116	0	0	0	23.4377	1.7813	0	0	23.5781	2.1788	0	0	24.4887	1.6649	0	0
f18	1.7183e+04	8169.7191	+	13.2965	2.1101	0	0	0	13.6744	2.5066	0	0	13.0940	2.5100	0	0	14.2692	2.7666	0	0
f19	875.5537	168.1187	+	133.4462	81.2072	0	0	0	123.0403	65.6581	0	0	134.4558	78.5449	0	0	151.1934	94.5234	0	0
f20	411.3699	27.4107	+	217.4900	2.2360	0	0	0	215.9934	2.4258	0	0	215.2000	3.0306	0	0	216.8278	2.6916	0	0
f21	8422.1911	4031.6448	+	953.8329	1559.1669	0	0	0	965.7101	1583.0733	0	0	1148.1378	1722.7757	0	0	1037.3169	1551.2075	0	0
f22	647.1978	30.0148	+	430.7109	7.5540	0	0	0	429.6142	7.0451	0	0	431.1878	4.9276	0	0	428.7289	6.0144	0	0
f23	708.3921	27.1088	+	505.9039	3.8658	0	0	0	506.5043	3.9147	0	0	507.2447	3.6806	0	0	506.4234	3.0773	0	0
f24	558.1646	12.9537	+	481.6379	3.8433	0	0	0	480.7105	2.3156	0	0	481.1730	3.2085	0	0	481.1785	3.2127	0	0
f25	3079.0504	331.0890	+	1147.8202	56.0881	0	0	0	1123.4536	70.3529	0	0	1111.9384	61.2884	0	0	1126.9762	50.5076	0	0
f26	609.0085	30.5432	+	510.9186	11.1726	0	0	0	512.4044	13.3798	0	0	513.3699	6.9740	0	0	512.4179	10.6421	0	0
f27	526.5269	20.1952	+	458.8487	0	0	0	0	458.8487	0	0	0	458.8487	0	0	0	458.8487	0	0	0
f28	1004.6185	185.2784	+	358.3330	18.4864	0	0	0	364.5377	13.4418	0	0	363.0278	16.0855	0	0	357.4481	13.1893	0	0
f29	3.8670e+06	8.7387e+05	+	6.0593e+05	3.7879e+04	0	0	0	6.1178e+05	3.6027e+04	0	0	6.1378e+05	3.3130e+04	0	0	6.0786e+05	3.8273e+04	0	0
+ / $\approx$ / -	29/0/0				0/27/2				1/22/6				1/20/8				1/23/5			

Table A9: Performance comparison of jSO with other algorithms on 50D. + means that jSO performs significantly better than the compared algorithm.

Func.	jSO+AGDE-MPP				jSO+SLDE				jSO+PRS				jSO+EAGDE				jSO			
	Mean	Std	sig		Mean	Std	sig		Mean	Std	sig		Mean	Std	sig		Mean	Std	sig	
f1	0	0	≈	0	0	0	≈	0	0	0	≈	0	0	0	≈	0	0	0	≈	0
f2	0	0	≈	0	0	0	≈	0	0	0	≈	0	0	0	≈	0	0	0	≈	0
f3	45.6174	37.3419	≈	66.5679	54.8045	≈	61.7864	47.5372	61.7864	47.5372	≈	54.8497	45.1760	≈	42.7048	37.3117	42.7048	37.3117	≈	37.3117
f4	15.3622	2.3348	≈	16.1593	3.2783	≈	17.1625	5.7003	17.1625	5.7003	≈	17.0337	3.2914	≈	16.6778	3.0663	16.6778	3.0663	≈	3.0663
f5	2.0428e-06	2.8104e-06	+	7.7120e-07	1.3971e-06	≈	7.6000e-08	1.2305e-07	7.6000e-08	1.2305e-07	≈	3.3804e-06	3.8098e-06	+	4.6960e-07	7.1185e-07	4.6960e-07	7.1185e-07	≈	7.1185e-07
f6	64.7646	3.3278	≈	66.2645	4.2389	≈	71.5340	4.6778	71.5340	4.6778	+	66.2313	2.7775	≈	67.0418	3.1429	67.0418	3.1429	≈	3.1429
f7	15.2826	2.3838	≈	17.7900	4.1555	≈	18.8771	4.0549	18.8771	4.0549	≈	17.7899	2.6685	≈	17.4339	3.3116	17.4339	3.3116	≈	3.3116
f8	0	0	≈	0	0	0	≈	0	0	0	≈	0	0	≈	0	0	0	0	≈	0
f9	2933.9102	354.6701	≈	2990.6051	414.5938	≈	4052.7558	277.9726	4052.7558	277.9726	+	3075.4880	331.7778	≈	3157.7247	324.3247	3157.7247	324.3247	≈	324.3247
f10	27.4594	3.4667	≈	27.1184	3.7646	≈	26.2946	3.2353	26.2946	3.2353	+	27.6773	3.8299	≈	28.9525	3.1583	28.9525	3.1583	≈	3.1583
f11	1671.8226	383.5354	≈	1855.1542	511.3165	≈	1885.6988	360.6361	1885.6988	360.6361	≈	1900.8875	433.9044	≈	1824.4905	461.1686	1824.4905	461.1686	≈	461.1686
f12	23.7344	16.8191	≈	25.1010	16.5127	≈	36.4336	23.7647	36.4336	23.7647	≈	32.4327	18.1177	≈	31.8991	21.5073	31.8991	21.5073	≈	21.5073
f13	25.4994	2.5727	≈	25.7805	2.3091	≈	24.2024	1.9568	24.2024	1.9568	≈	26.5504	2.4830	≈	25.3399	2.2669	25.3399	2.2669	≈	2.2669
f14	22.6737	2.3068	≈	23.1995	3.2014	≈	21.8718	1.8651	21.8718	1.8651	≈	25.3019	2.9210	≈	24.9593	2.9070	24.9593	2.9070	≈	2.9070
f15	445.2839	123.9434	≈	368.4019	127.8870	≈	436.4002	172.1896	436.4002	172.1896	≈	502.3542	105.6795	+	411.9340	131.0004	411.9340	131.0004	≈	131.0004
f16	299.5586	96.8226	≈	238.4152	127.5269	≈	321.1179	93.5213	321.1179	93.5213	+	316.4053	77.3379	≈	264.6026	81.8682	264.6026	81.8682	≈	81.8682
f17	23.4015	1.9287	≈	24.6807	1.9150	≈	23.3340	1.2461	23.3340	1.2461	≈	25.2760	2.2173	≈	24.3531	1.9339	24.3531	1.9339	≈	1.9339
f18	15.2052	2.5353	+	13.9888	3.0519	≈	13.0958	3.3676	13.0958	3.3676	≈	16.2019	2.2052	+	12.7583	1.7861	15.5696	81.0373	≈	81.0373
f19	95.9256	52.3994	≈	113.7997	98.0743	≈	185.1537	67.2380	185.1537	67.2380	+	168.1898	90.3477	≈	155.5696	81.0373	155.5696	81.0373	≈	81.0373
f20	216.5504	3.3253	≈	218.5272	2.5375	≈	217.6369	5.1152	217.6369	5.1152	≈	218.4149	3.7153	≈	217.3475	3.0116	217.3475	3.0116	≈	3.0116
f21	2079.1046	1809.4702	≈	1711.0631	1720.9508	≈	1593.6373	2229.9027	1593.6373	2229.9027	≈	1845.3187	1875.8805	≈	1460.2533	1858.3801	1460.2533	1858.3801	≈	1858.3801
f22	430.2123	7.9775	≈	441.9479	6.5353	+	432.0625	5.8133	432.0625	5.8133	≈	430.1371	6.2025	≈	432.0560	6.6002	432.0560	6.6002	≈	6.6002
f23	507.7962	3.9583	≈	518.5310	4.9477	+	507.9689	5.0849	507.9689	5.0849	≈	508.3338	3.5298	≈	507.0193	4.5214	507.0193	4.5214	≈	4.5214
f24	480.2534	1.5440e-02	≈	481.6459	3.8397	≈	480.7119	2.3240	480.7119	2.3240	≈	482.5707	4.7395	≈	482.9205	5.1481	482.9205	5.1481	≈	5.1481
f25	1156.4450	67.5403	≈	1231.7450	62.1570	+	1146.3877	59.3392	1146.3877	59.3392	≈	1144.9300	67.4544	≈	1151.0770	40.8196	1151.0770	40.8196	≈	40.8196
f26	511.1196	7.6137	≈	514.6412	11.7013	+	511.9310	11.8257	511.9310	11.8257	≈	510.4796	10.1846	≈	507.1303	10.9111	507.1303	10.9111	≈	10.9111
f27	460.8026	9.7692	≈	458.8487	0	≈	460.8026	9.7692	460.8026	9.7692	≈	460.8026	9.7692	≈	458.8487	0	458.8487	0	≈	0
f28	343.2069	12.1478	≈	351.4042	20.7107	≈	392.6863	21.5064	392.6863	21.5064	+	344.6227	14.2258	≈	363.2761	12.4372	363.2761	12.4372	≈	12.4372
f29	5.9475e+05	1.7825e+04	≈	6.1515e+05	4.8763e+04	≈	6.0240e+05	4.0171e+04	6.0240e+05	4.0171e+04	≈	6.0399e+05	2.9546e+04	≈	5.9977e+05	2.5149e+04	5.9977e+05	2.5149e+04	≈	2.5149e+04
+ / ≈ / -	2/21/6				4/22/3				5/21/3				3/25/1							

Table A10: Performance comparison of jSO with other algorithms on 100D. + means that jSO performs significantly better than the compared algorithm.

Func.	jSO+APDSDE				jSO+dynNP-DE				jSO+NLPSR				jSO+NLPSR-ver2				jSO+DSDCDE			
	Mean	Std	sig		Mean	Std	sig		Mean	Std	sig		Mean	Std	sig		Mean	Std	sig	
f1	3.0000e-08	6.0277e-08	+		0	0	≈		0	1.4224e-08	+		0	4.0571e-06	0	≈	2.2438e-02	4.4348e-02	+	
f2	1.5656e-06	2.3031e-06	-		1.1088e-06	1.2764e-06	-		8.8160e-07	1.5277e-06	-		4.0032e-06	4.0571e-06	+		124.1850	109.7019	+	
f3	198.3378	23.1052	≈		186.7962	30.5837	≈		195.4376	27.4309	≈		194.6005	20.5219	≈		201.4877	10.9546	+	
f4	41.4315	5.7573	≈		50.9132	5.8035	+		39.4004	4.8061	≈		36.6426	6.4843	+		83.5990	22.2493	+	
f5	9.4954e-05	2.1251e-04	≈		3.3970e-04	3.7321e-04	+		1.9054e-04	4.9844e-04	≈		2.8328e-04	5.5245e-04	≈		2.7970e-03	1.9658e-03	+	
f6	141.9505	5.6108	-		158.5464	4.9675	+		140.4639	5.7067	-		136.3430	6.7716	-		179.2037	19.1835	+	
f7	39.7587	5.9315	≈		49.1780	7.3406	+		37.4113	5.3591	-		35.0115	5.2572	-		88.3668	16.7985	+	
f8	4.3241e-02	9.4818e-02	≈		0.1801	0.2071	+		3.2230e-02	5.0908e-02	≈		0	0	-		0.7309	0.6065	+	
f9	8403.5622	635.8693	≈		1.0312e+04	437.4140	+		8289.3912	515.4253	-		9181.0315	553.0015	-		1.2731e+04	1355.6031	+	
f10	112.3891	40.9839	≈		141.8573	58.6725	+		104.5340	29.1647	≈		78.6136	17.3636	-		443.6386	166.1913	+	
f11	2.1228e+04	1.0095e+04	≈		2.4445e+04	1.2427e+04	+		1.9885e+04	7490.9450	≈		1.9680e+04	1.0275e+04	≈		8.3224e+04	2.7421e+04	+	
f12	192.3979	66.3425	+		197.3482	44.4885	+		173.1845	44.0878	≈		145.4975	33.3830	≈		851.2076	297.1923	+	
f13	64.4144	12.0565	≈		87.0244	19.9677	+		68.6656	16.1114	≈		55.5566	8.4749	≈		288.1441	68.8404	+	
f14	176.6007	34.4525	≈		195.7531	38.7628	+		151.1799	31.0017	≈		137.2397	35.0559	-		325.2886	69.5143	+	
f15	1677.6220	290.9059	≈		2016.6694	381.0808	+		1719.8973	291.7374	≈		1728.6221	327.4810	≈		3050.3209	493.3726	+	
f16	1132.5839	203.2866	-		1405.1008	212.0960	≈		1241.5002	210.0075	≈		1177.0822	233.4568	-		2351.7290	376.1421	+	
f17	188.0699	33.5550	≈		192.2350	38.5510	≈		182.1342	33.4708	≈		169.9714	34.0375	≈		1306.3777	1390.0493	+	
f18	109.5914	17.8126	≈		124.5947	15.1166	+		109.8406	18.4288	≈		98.3851	21.0912	≈		198.7469	40.9403	+	
f19	1264.9129	241.7437	≈		1575.2524	156.4415	+		1250.0623	179.7500	≈		1323.5916	244.8866	≈		2747.9962	402.7792	+	
f20	262.9197	5.8016	≈		266.4605	7.7806	≈		260.9023	5.6010	≈		256.6150	8.4156	-		313.6634	16.5895	+	
f21	9482.1877	798.5221	-		1.1318e+04	459.5074	+		9425.7947	456.3938	-		1.0339e+04	644.7054	≈		1.4337e+04	1249.9431	+	
f22	567.3680	14.7683	≈		570.1002	15.1339	≈		566.3284	15.3339	≈		570.2211	7.6596	≈		626.6314	17.2372	+	
f23	906.4526	6.5304	≈		902.1563	7.5288	≈		901.0392	6.3916	≈		899.0051	8.2383	≈		992.2961	26.7660	+	
f24	728.7311	39.6750	≈		725.6527	44.3796	≈		722.7817	39.1318	≈		719.8474	50.1031	≈		734.4191	35.2458	≈	
f25	3234.3300	73.0025	≈		3244.2611	80.0189	≈		3251.3206	86.2135	≈		3220.2382	96.3214	≈		4044.7549	217.3771	+	
f26	579.1593	20.2061	≈		579.0700	24.9911	≈		580.9586	20.2689	≈		579.6734	23.2495	≈		642.6701	22.5936	+	
f27	526.5946	26.1692	≈		530.3396	22.7610	≈		533.6753	25.2759	+		539.7663	32.6616	+		527.4203	25.5530	≈	
f28	1281.6984	158.5018	≈		1405.2543	210.4924	+		1278.1771	219.3669	≈		1184.3031	226.4419	≈		2348.0469	391.0307	+	
f29	2260.9078	95.9341	-		2385.8472	125.9175	≈		2341.0674	160.3095	≈		2273.2768	124.0745	≈		2578.7464	213.9584	+	
+ / ≈ / -			2 / 21 / 6				16 / 12 / 1				2 / 22 / 5				2 / 18 / 9				27 / 2 / 0	

Table A11: Performance comparison of jSO with other algorithms on 100D. + means that jSO performs significantly better than the compared algorithm.

Func.	jSO+FP				jSO+HPDE				jSO+HARD-DE				jSO+CS-DE				jSO+DPSHADE			
	Mean	Std	sig		Mean	Std	sig		Mean	Std	sig		Mean	Std	sig		Mean	Std	sig	
f1	9.0052e+06	2.7486e+06	+	≈	0	0	≈	≈	0	0	≈	≈	0	0	≈	≈	0	0	≈	≈
f2	5.3247e+04	8802.3728	+	≈	2.3672e-06	2.9037e-06	≈	≈	4.3532e-06	5.7364e-06	≈	≈	2.7352e-06	2.6086e-06	≈	≈	2.8384e-06	2.2178e-06	≈	≈
f3	278.2397	17.5988	+	≈	184.5952	24.6383	≈	≈	197.5685	13.9827	≈	≈	186.8588	31.7846	≈	≈	191.8270	30.8359	≈	≈
f4	475.2609	60.8170	+	≈	40.5652	6.5017	≈	≈	34.6514	5.8733	≈	≈	34.8151	7.4309	≈	≈	39.7484	6.6322	≈	≈
f5	3.5380	0.5434	+	≈	2.5799e-04	6.9449e-04	≈	≈	1.2480e-04	3.5420e-04	≈	≈	1.6530e-04	4.0308e-04	≈	≈	5.5390e-05	3.1291e-05	≈	≈
f6	642.9523	53.3218	+	≈	140.0329	6.9126	≈	≈	139.3099	3.8524	≈	≈	138.4196	5.3535	≈	≈	141.0402	6.9597	≈	≈
f7	482.7790	77.0783	+	≈	39.5554	5.6661	≈	≈	37.6899	5.9401	≈	≈	35.9189	4.9695	≈	≈	38.7020	6.1894	≈	≈
f8	95.5158	30.8471	+	≈	3.6079e-02	9.4416e-02	≈	≈	1.7906e-02	4.4764e-02	≈	≈	4.7089e-02	0.1261	≈	≈	1.7906e-02	4.4764e-02	≈	≈
f9	2.5025e+04	1020.4634	+	≈	9708.3006	646.8118	≈	≈	9628.2719	616.9033	≈	≈	9897.3596	654.5578	≈	≈	9641.8206	651.7985	≈	≈
f10	1283.1898	163.4072	+	≈	104.5132	28.8132	≈	≈	88.4349	30.1023	≈	≈	102.2263	29.9231	≈	≈	93.8768	25.8125	≈	≈
f11	9.7527e+06	4.0263e+06	+	≈	1.8788e+04	1.0283e+04	≈	≈	1.6954e+04	6709.4105	≈	≈	1.5498e+04	6608.2553	≈	≈	1.8333e+04	7332.6260	≈	≈
f12	3.3444e+04	5944.6407	+	≈	174.8442	39.9667	≈	≈	136.1187	36.9053	≈	≈	137.7086	37.6787	≈	≈	147.6078	33.6984	≈	≈
f13	791.1737	113.8663	+	≈	61.2408	10.3100	≈	≈	53.2866	7.2176	≈	≈	56.5896	9.0685	≈	≈	55.7715	10.5518	≈	≈
f14	1.6455e+04	4084.1357	+	≈	152.6572	39.0052	≈	≈	146.9426	26.6285	≈	≈	136.9866	28.6594	≈	≈	143.4076	33.4363	≈	≈
f15	4304.7287	593.9094	+	≈	1776.2721	265.4284	≈	≈	1844.5931	262.3836	≈	≈	1656.6190	292.4853	≈	≈	1796.6959	273.0690	≈	≈
f16	2911.7820	336.1231	+	≈	1244.1707	156.9156	≈	≈	1249.0655	287.5977	≈	≈	1200.0048	228.8336	≈	≈	1306.7588	272.9430	≈	≈
f17	2.3364e+04	6151.7884	+	≈	182.9694	41.6474	≈	≈	179.2357	36.1640	≈	≈	166.2252	35.3936	≈	≈	179.7516	29.9618	≈	≈
f18	7759.8037	4250.6535	+	≈	99.7199	22.0137	≈	≈	91.3921	15.8559	≈	≈	87.3598	20.1770	≈	≈	103.1350	15.7823	≈	≈
f19	3119.9339	340.2396	+	≈	1392.9679	318.6805	≈	≈	1344.9221	212.9396	≈	≈	1332.8580	205.0433	≈	≈	1325.8403	267.1815	≈	≈
f20	686.1083	83.3039	+	≈	262.7314	5.6621	≈	≈	258.6601	5.9231	≈	≈	255.5065	6.3582	≈	≈	262.3803	8.5824	≈	≈
f21	2.6381e+04	1134.6417	+	≈	1.0458e+04	626.3567	≈	≈	1.0824e+04	761.7255	≈	≈	1.0339e+04	2267.0192	≈	≈	1.0070e+04	2156.2394	≈	≈
f22	1014.1402	78.4192	+	≈	571.9262	12.9669	≈	≈	570.3082	14.2440	≈	≈	569.4101	10.0994	≈	≈	571.7824	11.1708	≈	≈
f23	1352.5211	74.9923	+	≈	896.2882	7.2350	≈	≈	902.0401	7.6418	≈	≈	899.9642	6.7179	≈	≈	901.7375	6.2682	≈	≈
f24	862.1031	33.5748	+	≈	731.3078	40.9726	≈	≈	743.0685	42.9956	≈	≈	732.4849	34.7966	≈	≈	734.0573	37.6685	≈	≈
f25	7197.2949	701.7365	+	≈	3217.0262	81.2718	≈	≈	3257.5044	86.2789	≈	≈	3215.5469	88.4543	≈	≈	3230.0317	74.5060	≈	≈
f26	671.2340	30.5458	+	≈	582.1033	17.9139	≈	≈	584.4882	21.0316	≈	≈	586.0847	22.8809	≈	≈	586.3314	35.0266	≈	≈
f27	639.3029	34.2838	+	≈	530.4429	26.1549	≈	≈	516.7899	16.7618	≈	≈	530.4228	24.5862	≈	≈	527.9197	28.8106	≈	≈
f28	3366.3916	384.8760	+	≈	1307.5620	214.1985	≈	≈	1232.8255	185.5356	≈	≈	1247.8242	214.0099	≈	≈	1328.3308	210.5044	≈	≈
f29	1.6977e+05	5.2742e+04	+	≈	2301.9143	118.2183	≈	≈	2310.5734	99.1250	≈	≈	2294.9451	143.9851	≈	≈	2278.5839	102.6015	≈	≈
+ / ≈ / -	29/0/0				0/27/2				0/23/6				0/23/6				0/27/2			

Table A12: Performance comparison of jSO with other algorithms on 100D. + means that jSO performs significantly better than the compared algorithm.

Func.	jSO+AGDE-MPP				jSO+SLDE				jSO+PRS				jSO+EAGDE				jSO			
	Mean	Std	sig		Mean	Std	sig		Mean	Std	sig		Mean	Std	sig		Mean	Std		
f1	6.2641e-04	1.2812e-03	+		0	0	$\approx$		0	0	$\approx$		2.2880e-07	5.5269e-07	+		0	0		
f2	2.2716e-06	1.9568e-06	$\approx$		2.1396e-06	2.0796e-06	$\approx$		1.7353e-05	1.5490e-05	+		8.1840e-07	1.3821e-06	-		1.8924e-06	1.7066e-06		
f3	197.0933	20.4889	$\approx$		181.3783	32.3096	$\approx$		202.9235	17.2287	+		192.6534	32.7005	$\approx$		192.1875	22.7492		
f4	34.7440	5.8364	-		35.8277	5.0864	-		56.3677	12.2672	+		39.5198	4.7848	$\approx$		42.1795	6.0242		
f5	1.1668e-04	2.0566e-04	$\approx$		2.4467e-04	5.4093e-04	$\approx$		4.7452e-05	7.6034e-05	-		1.8358e-04	3.3317e-04	$\approx$		4.6008e-04	8.2696e-04		
f6	137.7519	5.3878	-		138.4092	6.2056	-		167.7576	12.9395	+		141.8603	4.7517	-		148.2234	8.7664		
f7	33.5102	4.0387	-		35.7442	5.2827	-		53.9343	14.8318	+		39.5596	4.6427	$\approx$		42.6103	6.2833		
f8	1.4325e-02	5.5910e-02	$\approx$		9.3911e-02	0.1987	-		2.8916e-02	9.6950e-02	+		6.1414e-02	0.1402	$\approx$		7.6006e-02	0.1628		
f9	8686.8977	537.5051	-		8649.4314	755.4090	-		1.2185e+04	635.0660	+		8654.2399	487.0493	-		9682.9285	559.0376		
f10	97.6366	34.0409	$\approx$		112.0963	38.8107	$\approx$		91.9316	28.0956	$\approx$		118.4787	26.3976	$\approx$		110.4749	35.1581		
f11	2.2787e+04	8514.7619	+		1.9600e+04	9782.7560	$\approx$		1.5102e+04	6941.3365	$\approx$		2.1757e+04	1.1797e+04	$\approx$		1.7567e+04	8198.7151		
f12	159.0578	39.1656	$\approx$		146.7237	44.0858	$\approx$		126.2795	33.3437	-		184.2335	42.0641	+		153.5380	32.9257		
f13	54.4567	9.4208	-		63.8907	9.8828	$\approx$		53.1493	8.9338	-		73.7548	13.9671	+		61.7553	12.4284		
f14	138.3706	31.4417	-		160.4031	47.4080	$\approx$		157.8775	41.0271	$\approx$		184.1514	51.0230	$\approx$		158.0054	33.8829		
f15	1620.8590	376.3405	$\approx$		1738.9241	391.8130	$\approx$		2081.5938	437.3678	+		1777.0001	291.1793	$\approx$		1770.3289	326.2236		
f16	1177.4544	250.7621	-		1111.4077	244.7007	-		1459.3192	306.0739	$\approx$		1203.6706	210.8822	-		1319.4011	196.5215		
f17	179.4886	35.8420	$\approx$		187.1989	25.3928	$\approx$		170.2085	37.5967	$\approx$		183.5809	35.7232	$\approx$		180.8979	42.8708		
f18	95.0413	19.3790	$\approx$		105.0692	22.3730	$\approx$		86.9863	17.6647	-		114.8325	22.0702	+		101.1701	17.9910		
f19	1351.7396	242.1094	$\approx$		1351.3988	327.4814	$\approx$		1758.1428	380.2910	+		1289.6234	224.6725	$\approx$		1368.5323	242.2703		
f20	257.4899	5.7633	-		263.7025	5.5434	$\approx$		270.7570	11.5076	+		261.9272	6.0993	$\approx$		263.7504	6.5101		
f21	9470.3001	820.9147	-		9338.3493	794.3561	-		1.3147e+04	874.8553	+		9442.5012	731.4970	-		1.0547e+04	818.2048		
f22	571.3787	11.6424	$\approx$		579.9356	13.0069	+		572.8999	9.7314	$\approx$		574.6802	13.4549	+		565.5059	12.3325		
f23	900.0682	8.6784	$\approx$		938.4594	9.4033	+		899.5012	7.7864	$\approx$		901.5048	9.1255	$\approx$		901.5313	9.7192		
f24	723.5732	37.0255	$\approx$		744.1111	32.9772	$\approx$		731.2603	40.5158	$\approx$		730.3599	31.0234	$\approx$		724.3528	42.8851		
f25	3259.8942	91.9930	$\approx$		3490.2979	110.4635	+		3243.0676	89.9581	$\approx$		3226.1392	87.9234	$\approx$		3224.9706	90.0328		
f26	581.5151	22.3337	$\approx$		592.2363	24.1630	$\approx$		581.9854	18.8088	$\approx$		588.4810	22.7912	$\approx$		579.3070	26.2424		
f27	528.2171	22.7645	$\approx$		527.2278	23.8039	$\approx$		523.4791	25.7636	$\approx$		533.9809	27.3054	+		519.6381	24.6112		
f28	1226.9736	204.4672	$\approx$		1236.2872	200.7299	$\approx$		1286.1886	231.1021	$\approx$		1311.1424	232.4750	$\approx$		1248.7132	221.5815		
f29	2294.6525	126.7202	$\approx$		2284.2335	134.6136	$\approx$		2288.7731	88.9879	$\approx$		2321.1806	152.5922	$\approx$		2333.5068	142.3203		
+ / $\approx$ / -	2 / 18 / 9				3 / 20 / 6				10 / 15 / 4				6 / 18 / 5							

Table A13: Performance comparison of jSO with other algorithms on 10D. + means that jSO performs significantly better than the compared algorithm.

Func.	jSO-wo-Half			jSO-wo-Red			DE			jSO		
	Mean	Std	sig	Mean	Std	sig	Mean	Std	sig	Mean	Std	sig
f1	0	0	$\approx$	0	0	$\approx$	0	0	$\approx$	0	0	0
f2	0	0	$\approx$	0	0	$\approx$	0	0	$\approx$	0	0	0
f3	0	0	$\approx$	0	0	$\approx$	1.0264e-03	5.1318e-03	$\approx$	0	0	0
f4	3.2104	0.9876	+	3.2540	0.7950	+	23.5513	3.3777	+	1.5521	0.8164	0.8164
f5	0	0	$\approx$	0	0	$\approx$	0	0	$\approx$	0	0	0
f6	13.3960	0.9125	+	13.5581	1.1595	+	33.7499	3.6944	+	11.9474	0.5542	0.5542
f7	3.3695	0.9678	+	3.3902	1.2472	+	23.3779	3.3567	+	1.9899	0.8124	0.8124
f8	0	0	$\approx$	0	0	$\approx$	0	0	$\approx$	0	0	0
f9	205.7335	102.6370	+	185.7938	99.3058	+	972.2365	177.2143	+	32.4858	46.7644	46.7644
f10	0.4255	0.7362	+	0.5191	0.7013	+	3.9883e-02	0.1990	+	0	0	0
f11	19.7449	44.4909	$\approx$	19.4862	44.7486	$\approx$	0.3423	0.3748	$\approx$	0.3580	0.1412	0.1412
f12	4.5621	2.2665	+	3.9105	2.4806	$\approx$	3.5732	2.9019	$\approx$	3.2180	2.3938	2.3938
f13	2.3414	2.0891	+	2.0736	2.2621	+	0.2138	0.8639	+	3.9798e-02	0.1990	0.1990
f14	0.2441	0.2243	$\approx$	0.2630	0.2338	$\approx$	2.9625e-03	4.6535e-03	-	0.2653	0.2056	0.2056
f15	1.3935	0.5958	+	1.4190	0.6250	+	0.3015	0.1617	-	0.5638	0.2759	0.2759
f16	3.0358	1.6401	+	4.7501	1.8379	+	5.7228	6.0687	+	0.4262	0.2664	0.2664
f17	0.2267	0.2267	$\approx$	0.1938	0.2336	$\approx$	1.3213e-02	4.5822e-02	-	0.2865	0.2186	0.2186
f18	7.1075e-02	0.1238	+	8.1968e-02	0.1604	+	6.2545e-03	1.1890e-02	$\approx$	9.3511e-03	1.2751e-02	1.2751e-02
f19	0.3899	0.2456	$\approx$	0.6262	0.5015	+	0.2123	0.1738	-	0.3871	0.1632	0.1632
f20	154.5007	53.4494	$\approx$	142.4610	50.3094	$\approx$	169.4712	62.9333	+	145.2757	59.5918	59.5918
f21	100.0000	0	$\approx$	98.0827	9.5864	$\approx$	96.1594	20.0346	+	100.0000	0	0
f22	303.9664	1.6693	+	303.7401	1.9531	+	310.6149	5.9574	+	300.6974	1.4499	1.4499
f23	284.9942	94.3776	+	294.2634	86.5418	+	256.4229	120.2072	$\approx$	264.9539	104.9940	104.9940
f24	410.6696	20.8042	$\approx$	412.4855	21.6035	$\approx$	399.8407	9.5747	$\approx$	408.8673	19.7697	19.7697
f25	300.0000	0	$\approx$	300.0000	0	$\approx$	300.0000	0	$\approx$	300.0000	0	0
f26	389.4155	0.2092	$\approx$	389.3745	0.2348	$\approx$	390.0890	1.6754	$\approx$	389.4360	0.1917	0.1917
f27	348.7684	114.1547	$\approx$	370.3441	127.9308	$\approx$	300.0000	0	$\approx$	324.9457	86.3395	86.3395
f28	245.5835	4.4005	+	245.8515	5.6639	+	240.1197	4.3986	+	235.1833	3.7476	3.7476
f29	394.5229	4.4108e-02	$\approx$	398.3705	13.3296	$\approx$	396.8685	4.8002	+	394.5191	4.6988e-02	4.6988e-02
+ / $\approx$ / -	13/16/0			13/16/0			12/13/4					

Table A14: Performance comparison of jSO with other algorithms on 30D. + means that jSO performs significantly better than the compared algorithm.

Func.	jSO-wo-Half			jSO-wo-Red			DE			jSO		
	Mean	Std	sig	Mean	Std	sig	Mean	Std	sig	Mean	Std	Std
f1	0	0	$\approx$	0	0	$\approx$	0	0	$\approx$	0	0	0
f2	0	0	$\approx$	0	0	$\approx$	12.125	9.0058	+	0	0	0
f3	58.7838	1.1112	$\approx$	47.9943	23.3587	$\approx$	58.7838	1.1112	$\approx$	58.5616	0	0
f4	42.3114	3.9720	+	19.6316	2.9399	+	179.1810	8.5977	+	8.5503	1.9831	1.9831
f5	0	0	$\approx$	1.1080e-07	2.9921e-07	$\approx$	0	0	$\approx$	1.1200e-08	3.8764e-08	3.8764e-08
f6	80.1962	8.3836	+	49.5137	2.5127	+	209.6285	9.8081	+	38.9737	1.6484	1.6484
f7	41.5600	4.4120	+	20.0750	3.7328	+	181.2514	7.9663	+	9.1448	1.8277	1.8277
f8	0	0	$\approx$	0	0	$\approx$	0	0	$\approx$	0	0	0
f9	3659.9329	262.3260	+	2302.9965	238.3563	+	6825.4268	211.4563	+	1464.8674	248.8887	248.8887
f10	17.6634	20.1117	+	24.2864	26.3800	+	50.1288	14.6796	+	2.8312	2.0799	2.0799
f11	270.3686	182.4955	$\approx$	816.0069	332.5200	+	7149.2135	4524.5264	+	216.3273	167.1318	167.1318
f12	34.7400	6.8018	+	48.7872	37.3129	+	79.8439	9.0122	+	17.0228	5.2018	5.2018
f13	31.7282	2.1805	+	27.6131	2.8100	+	63.6367	4.3095	+	20.0149	5.4341	5.4341
f14	12.0289	1.8590	+	10.9664	7.4443	+	35.0112	9.1651	+	1.3631	0.9776	0.9776
f15	333.4725	132.5508	+	292.9411	182.1942	+	735.2669	445.3031	+	66.9240	72.1669	72.1669
f16	95.5719	10.8534	+	59.7337	14.1707	+	71.1435	7.2400	+	36.3462	6.8523	6.8523
f17	22.4583	1.5717	+	26.8790	5.1427	+	36.0905	4.1997	+	20.9854	0.5952	0.5952
f18	15.9150	1.5034	+	14.0220	4.0397	+	17.0103	6.3938	+	4.5346	1.5415	1.5415
f19	116.3151	24.9918	+	67.9031	25.6891	+	25.2129	15.3827	$\approx$	29.3567	5.6791	5.6791
f20	240.6282	4.4580	+	220.6066	3.0846	+	368.5389	7.5346	+	208.5554	2.3110	2.3110
f21	100.0000	0	$\approx$	100.0000	0	$\approx$	100.0000	0	$\approx$	100.0000	0	0
f22	384.1308	6.2138	+	365.7029	6.4735	+	519.9889	10.5863	+	349.6782	2.9115	2.9115
f23	455.1750	6.3300	+	435.3637	4.3389	+	584.8942	10.2894	+	425.8976	1.7728	1.7728
f24	386.6998	9.2425e-03	$\approx$	386.7364	3.2877e-02	+	386.7411	2.1635e-02	+	386.6972	8.4031e-03	8.4031e-03
f25	1272.6801	107.8685	+	1053.1830	170.5028	+	2258.4716	608.9886	+	944.7799	45.1769	45.1769
f26	497.2984	7.6686	$\approx$	501.2256	8.8659	+	490.6236	8.3162	$\approx$	493.9173	7.8760	7.8760
f27	331.9130	52.2300	$\approx$	338.6329	58.8047	$\approx$	312.3943	34.2560	$\approx$	312.8219	35.4825	35.4825
f28	528.1840	22.6216	+	481.2342	19.5066	+	540.3465	121.3232	+	425.6575	27.0707	27.0707
f29	1978.1987	23.3660	+	2034.5928	92.8157	+	1987.7943	38.5875	+	1971.9225	11.0916	11.0916
+ / $\approx$ / -	19/10/0			22/7/0			21/8/0					

Table A15: Performance comparison of jSO with other algorithms on 50D. + means that jSO performs significantly better than the compared algorithm.

Func.	jSO-wo-Half				jSO-wo-Red				DE				jSO			
	Mean	Std	sig		Mean	Std	sig		Mean	Std	sig		Mean	Std	sig	
f1	0	0	≈		0	0	≈		77.1775	238.7580	+		0	0	+	
f2	0	0	≈		0	0	≈		5.6631e+04	1.2583e+04	+		0	0	+	
f3	53.9538	43.5601	≈		59.5648	51.6250	≈		81.1508	44.8551	+		42.7048	37.3117	+	
f4	118.0341	12.8057	+		44.4716	8.0144	+		354.1972	12.1989	+		16.6778	3.0663	+	
f5	1.8480e-07	4.0006e-07	-		1.9737e-05	3.2086e-05	+		1.0560e-06	3.4705e-06	-		4.6960e-07	7.1185e-07	+	
f6	179.0148	13.1749	+		98.0494	7.1572	+		402.8993	13.2481	+		67.0418	3.1429	+	
f7	114.9193	10.6541	+		46.3229	6.7775	+		356.6496	11.9238	+		17.4339	3.3116	+	
f8	0	0	≈		0.6447	0.4781	+		1.4325e-02	3.3498e-02	+		0	0	+	
f9	7578.7149	383.9445	+		4298.3757	309.3364	+		1.2895e+04	315.9155	+		3157.7247	324.3247	+	
f10	58.2705	8.0357	+		77.0509	18.3152	+		130.6679	36.9136	+		28.9525	3.1583	+	
f11	1611.6905	384.4067	≈		7578.3041	6673.3555	+		6.3963e+04	4.8924e+04	+		1824.4905	461.1686	+	
f12	119.4642	54.5813	+		129.6314	87.8683	+		281.1683	89.6075	+		31.8991	21.5073	+	
f13	60.3139	6.5283	+		58.3999	13.3993	+		125.2950	9.1703	+		25.3399	2.2669	+	
f14	31.3203	11.1949	+		81.3189	42.0330	+		109.6400	10.3061	+		24.9593	2.9070	+	
f15	1110.0676	184.9640	+		767.7493	168.5745	+		2294.3239	768.0798	+		411.9340	131.0004	+	
f16	745.8115	111.3078	+		546.0360	99.1880	+		1134.0312	473.1206	+		264.6026	81.8682	+	
f17	28.6203	5.7473	+		150.2571	80.6727	+		528.1716	370.1393	+		24.3531	1.9339	+	
f18	33.6432	2.3328	+		59.2202	21.9595	+		61.0342	11.1425	+		12.7583	1.7861	+	
f19	575.9835	125.0489	+		318.7817	113.7147	+		843.8771	437.9974	+		155.5696	81.0373	+	
f20	322.7840	10.5230	+		243.3978	6.0219	+		554.7086	10.9797	+		217.3475	3.0116	+	
f21	5068.9473	3539.6456	+		4296.5115	1615.0068	+		7388.0769	6598.7618	+		1460.2533	1858.3801	+	
f22	542.5332	14.7714	+		465.8653	11.9974	+		766.3361	18.9653	+		432.0560	6.6002	+	
f23	616.5303	23.1328	+		539.4605	8.7654	+		835.3793	14.6299	+		507.0193	4.5214	+	
f24	485.1425	18.4693	≈		513.7100	40.4150	+		499.5745	31.2140	+		482.9205	5.1481	+	
f25	2224.2955	253.4213	+		1543.8300	129.7585	+		4076.8170	646.3637	+		1151.0770	40.8196	+	
f26	516.6807	10.3295	+		563.7944	45.3581	+		509.3493	10.9505	≈		507.1303	10.9111	+	
f27	462.7564	13.5248	≈		477.0630	23.0145	+		464.8470	16.5881	≈		458.8487	0	+	
f28	648.2222	46.2663	+		494.4117	74.0321	+		572.2938	258.2466	+		363.2761	12.4372	+	
f29	6.2126e+05	3.8218e+04	+		6.7480e+05	8.8104e+04	+		5.8613e+05	1.5331e+04	≈		5.9977e+05	2.5149e+04	+	
+ / ≈ / -	21 / 7 / 1				26 / 3 / 0				25 / 3 / 1							

Table A16: Performance comparison of jSO with other algorithms on 100D. + means that jSO performs significantly better than the compared algorithm.

Func.	jSO-wo-Half			jSO-wo-Red			DE			jSO		
	Mean	Std	sig	Mean	Std	sig	Mean	Std	sig	Mean	Std	sig
f1	0	0	≈	0	0	≈	6297.5872	7831.9493	+	0	0	+
f2	6.9200e-04	5.0627e-04	+	2.8160e-07	9.2620e-07	-	3.1040e+05	2.4113e+04	+	1.8924e-06	1.7066e-06	+
f3	188.4600	26.8218	≈	30.1233	57.9608	-	209.5581	35.0422	+	192.1875	22.7492	+
f4	377.9587	32.2184	+	186.6785	32.9881	+	810.3526	22.6946	+	42.1795	6.0242	+
f5	1.8646e-04	5.3596e-04	-	0.1668	0.1515	+	5.6554e-04	9.1440e-04	+	4.6008e-04	8.2696e-04	+
f6	500.7102	22.0711	+	332.2210	37.3976	+	928.8761	23.6022	+	148.2234	8.7664	+
f7	384.5640	34.0408	+	177.3998	21.9437	+	820.0319	17.8679	+	42.6103	6.2833	+
f8	0.1413	0.1980	≈	97.6732	145.9116	+	5.1992	4.5353	+	7.6006e-02	0.1628	+
f9	2.0348e+04	750.0471	+	1.1036e+04	560.9008	+	2.9428e+04	445.1559	+	9682.9285	559.0376	+
f10	133.4970	43.2786	≈	679.6169	174.1829	+	594.3109	47.1077	+	110.4749	35.1581	+
f11	2.6805e+04	9865.3542	+	2.4640e+04	1.2093e+04	+	2.3340e+05	6.5851e+04	+	1.7567e+04	8198.7151	+
f12	175.1451	48.9585	≈	955.6968	824.1515	+	2991.2022	2776.1175	+	153.5380	32.9257	+
f13	86.9437	44.5164	+	615.3115	127.5950	+	8384.7757	5890.2600	+	61.7553	12.4284	+
f14	171.9190	48.0516	≈	382.8687	92.1207	+	1751.4844	2841.7035	+	158.0054	33.8829	+
f15	4391.8185	372.5279	+	2629.4526	388.8287	+	7583.3327	223.0941	+	1770.3289	326.2236	+
f16	2753.5553	204.6202	+	1841.0393	293.4658	+	4480.2291	576.2551	+	1319.4011	196.5215	+
f17	221.2151	53.7506	+	989.1538	896.6723	+	1.1745e+05	4.4592e+04	+	180.8979	42.8708	+
f18	117.0635	22.5321	+	283.4901	66.0149	+	2110.0363	2409.4343	+	101.1701	17.9910	+
f19	2939.3069	231.3056	+	1818.8639	211.8997	+	4553.5300	218.6408	+	1368.5323	242.2703	+
f20	611.8647	30.6451	+	393.7816	27.9329	+	1042.7436	19.3117	+	263.7504	6.5101	+
f21	2.1363e+04	833.6020	+	1.2493e+04	654.9590	+	3.0203e+04	605.8924	+	1.0547e+04	818.2048	+
f22	800.2853	161.9885	+	732.3375	30.6168	+	722.0095	244.3876	+	565.5059	12.3325	+
f23	961.3863	122.3517	+	1064.9069	47.3823	+	1454.4970	301.6624	+	901.5313	9.7192	+
f24	736.7017	34.7614	≈	742.8394	52.3235	≈	734.3601	47.1616	≈	724.3528	42.8851	+
f25	3708.7443	1091.0251	+	5293.5838	468.1913	+	4799.6004	2637.6651	+	3224.9706	90.0328	+
f26	598.1489	25.7736	+	680.3636	27.3468	+	594.9174	16.9963	+	579.3070	26.2424	+
f27	532.1355	35.7609	≈	498.6878	56.9249	≈	558.0172	27.5655	+	519.6381	24.6112	+
f28	2939.0127	287.8382	+	2241.9291	368.6846	+	4470.9011	1180.9155	+	1248.7132	221.5815	+
f29	2369.2049	140.0810	≈	3381.1290	2517.7727	+	3949.8511	1967.5551	+	2333.5068	142.3203	+
+ / ≈ / -	19/9/1			24/3/2			28/1/0					

Table A17: Algorithms' ranks in 10D.

Algorithm	MeanRank	Rank
jSO-wo-Half	2.7586207	3
jSO-wo-Red	2.9137931	4
DE	2.3965517	2
jSO	1.9310345	1
$\chi^2 = 12.2936, p = 0.0064421$		

Table A18: Algorithms' ranks in 30D.

Algorithm	MeanRank	Rank
jSO-wo-Half	2.7241379	3
jSO-wo-Red	2.5000000	2
DE	3.3793103	4
jSO	1.3965517	1
$\chi^2 = 40.3882, p = 8.8156\text{e-}09$		

Table A19: Algorithms' ranks in 50D.

Algorithm	MeanRank	Rank
jSO-wo-Half	2.5000000	2
jSO-wo-Red	2.6551724	3
DE	3.6551724	4
jSO	1.1896552	1
$\chi^2 = 55.2278, p = 6.1394\text{e-}12$		

Table A20: Algorithms' ranks in 100D.

Algorithm	MeanRank	Rank
jSO-wo-Half	2.5517241	2
jSO-wo-Red	2.5862069	3
DE	3.6551724	4
jSO	1.2068966	1
$\chi^2 = 53.2238, p = 1.6426\text{e-}11$		

Table A21: Results for real-world problem rwp1.

Algorithm	Best	Worst	Median	Mean	Std
jSO+AGDE-MPP	0.0000	10.9423	0.0000	0.8444	2.9247
jSO+APDSDE	0.0000	10.1675	0.0000	0.4067	2.0335
jSO+CS-DE	0.0000	0.0000	0.0000	0.0000	0.0000
jSO+DPSHADE	0.0000	10.1675	0.0000	0.4067	2.0335
jSO+DSDCDE	0.0000	13.2286	0.4879	4.2609	5.4187
jSO+EAGDE	0.0000	10.1675	0.0000	0.4067	2.0335
jSO+FP	13.5528	22.9329	18.0797	17.8412	3.1144
jSO+HARD-DE	0.0000	10.1675	0.0000	0.4067	2.0335
jSO+HPDE	0.0000	0.0000	0.0000	0.0000	0.0000
jSO+NLPSR	0.0000	0.0000	0.0000	0.0000	0.0000
jSO+NLPSR-ver2	0.0000	10.0796	0.0000	0.4032	2.0159
jSO+PRS	0.0000	0.0000	0.0000	0.0000	0.0000
jSO+dynNP-DE	0.0000	10.1675	0.0000	0.4067	2.0335
jSO+SLDE	0.0000	11.2072	0.0000	0.8550	2.9630
jSO	0.0000	0.0000	0.0000	0.0000	0.0000
jSO-wo-Red	0.0000	10.1675	0.0308	1.6607	3.2740
jSO-wo-Half	0.0000	11.5757	0.0000	1.2568	3.4818
DE	0.0000	0.0000	0.0000	0.0000	0.0000

Table A22: Results for real-world problem rwp2.

Algorithm	Best	Worst	Median	Mean	Std
jSO+AGDE-MPP	-28.4222	-25.4382	-27.1433	-27.1679	0.6654
jSO+APDSDE	-28.4032	-26.106	-27.2133	-27.1227	0.6461
jSO+CS-DE	-28.2367	-24.3886	-25.8302	-25.8955	0.7442
jSO+DPSHADE	-27.2193	-24.4337	-25.8634	-25.8319	0.6946
jSO+DSDCDE	-16.849	-4.00683	-8.79793	-9.00533	3.3681
jSO+EAGDE	-28.4223	-26.3175	-27.278	-27.2995	0.5924
jSO+FP	-20.4595	-13.9379	-17.2478	-17.155	1.8150
jSO+HARD-DE	-27.4041	-24.6812	-25.847	-25.865	0.5978
jSO+HPDE	-27.2179	-24.6182	-25.9998	-25.9545	0.6208
jSO+NLPSR	-28.4166	-26.4709	-27.3349	-27.3203	0.4919
jSO+NLPSR-ver2	-27.8716	-25.0178	-26.0068	-26.2352	0.7285
jSO+PRS	-25.8833	-21.9485	-23.9669	-23.917	0.8841
jSO+dynNP-DE	-27.5428	-25.6162	-26.2716	-26.4868	0.6880
jSO+SLDE	-27.3288	-24.9546	-26.2607	-26.2822	0.6367
jSO	-26.9955	-24.96	-25.7099	-25.9331	0.5629
jSO-wo-Red	-25.2622	-23.0344	-23.8332	-23.9828	0.5588
jSO-wo-Half	-21.3602	-18.6522	-19.5171	-19.6478	0.6953
DE	-15.5024	-10.2512	-11.9645	-12.2473	1.2085

Appendix B Collection of the references

Tables B23 and B24 show collections of the references with the LPSR and non-LPSR mechanisms, respectively. They correspond to Section 4.

Table B23: A collection of the references with the LPSR mechanism. Publication type T is either J (journal paper) or C (conference paper or book-chapter).

Reference	T	Reference	T	Reference	T
Biswas et al. (2017c)	J	Mohamed et al. (2017)	C	Hraiz et al. (2017)	C
Awad et al. (2017)	C	Biswas et al. (2017b)	C	Biswas et al. (2017a)	C
Biswas et al. (2017d)	C	Lin et al. (2017)	C	Viktorin et al. (2017)	C
Zamuda (2017)	C	Pholdee and Bureerat (2018)	J	Piotrowski (2018)	J
Sun et al. (2018)	J	Viktorin et al. (2018b)	J	Bujok and Polaková (2018)	C
Biswas et al. (2018b)	C	Biswas et al. (2018a)	C	Hamdi et al. (2018)	C
Viktorin et al. (2018a)	C	Biswas et al. (2019)	J	Choi et al. (2019)	J
Goudos et al. (2019)	J	Hadi et al. (2019)	J	Hamdi et al. (2019)	J
Meng et al. (2019b)	J	Sarajlić et al. (2019)	J	Sato et al. (2019)	J
Tian and Gao (2019)	J	Viktorin et al. (2019b)	J	Viktorin et al. (2019a)	J
Xia et al. (2019)	J	Zamuda and Sosa (2019b)	J	Bujok and Zamuda (2019)	C
Liu (2019)	C	Salgotra et al. (2019)	C	Sallam et al. (2019)	C
Yeh et al. (2019)	C	Zamuda (2019b)	C	Zamuda (2019a)	C
Zamuda and Sosa (2019a)	C	Bujok (2020)	J	Cheng et al. (2020)	J
Hao et al. (2020)	J	Hoang and Nguyen (2020)	J	Hoang (2020)	J
Huang et al. (2020)	J	Nasir et al. (2020)	J	Song et al. (2020)	J
Sun et al. (2020)	J	Wei et al. (2020)	J	Biswas and Suganthan (2020)	C
Stanovov et al. (2020)	C	El-Abd et al. (2020)	C	Mousavirad and Rahnamayan (2020)	C
Aranguren et al. (2021)	J	Bujok (2021)	J	Chen et al. (2021c)	J
Chen et al. (2021b)	J	Chen et al. (2021a)	J	Dinh et al. (2021)	J
Huang et al. (2021)	J	Koger and Uymaz (2021)	J	Nguyen et al. (2021)	J
Poláková and Bujok (2021)	J	Refaat et al. (2021)	J	Rizk-Allah and El-Fergany (2021)	J
Tan et al. (2021)	J	Song et al. (2021)	J	Yao et al. (2021)	J
Zuo et al. (2021)	J	Hadi et al. (2021)	C	Biswas et al. (2021)	C
Brest et al. (2021)	C	Yang et al. (2021)	C	Chang et al. (2022)	J
Ghosh et al. (2022)	J	Ji et al. (2022)	J	Li et al. (2022a)	J
Li et al. (2022c)	J	Mousavirad et al. (2022)	J	Ozkaya et al. (2022)	J
Stanovov et al. (2022)	J	Tang and Lee (2022)	J	Wang et al. (2022)	J
Zhang et al. (2022)	J	Bujok and Kolenovsky (2022)	C	Salgotra et al. (2022)	C
Lei et al. (2023)	J	Zhang and Meng (2023)	J	Ahmed et al. (2023)	J
Aly et al. (2023)	J	Cherifi et al. (2023)	J	Emami et al. (2023)	J
Gu et al. (2023b)	J	Gu et al. (2023a)	J	Li et al. (2023a)	J
Li et al. (2023c)	J	Meng and Zhang (2023a)	J	Stanovov and Semnkin (2023)	J
Tan et al. (2023)	J	Tang et al. (2023)	J	Yang et al. (2023)	J
Zhou et al. (2023)	J	Dong et al. (2023)	C	de Rezende Freitas et al. (2023)	C
Cheng et al. (2024)	J	Cui et al. (2024)	J	Huang et al. (2024)	J
Khodkari et al. (2024)	J	Stanovov and Semnkin (2024a)	J	Sun et al. (2024)	J
Visutarrom and Chiang (2024)	J	Wang et al. (2024)	J	Yang et al. (2024a)	J
Yang et al. (2024c)	J	Yang et al. (2024d)	J	Zhou et al. (2024)	J
Stanovov and Semnkin (2024b)	C	Blaga et al. (2025)	J	Chen et al. (2025)	J
Durmuş et al. (2025)	J	Fu et al. (2025)	J	Gehlert et al. (2025)	J
Gupta et al. (2025)	J	Kostovska et al. (2025)	J	Kou et al. (2025)	J
Ma et al. (2025)	J	Naas et al. (2025)	J	Sallam et al. (2025)	J
Song et al. (2025)	J	Wen et al. (2025)	J	Yadav et al. (2025)	J
Yildirim et al. (2025)	J	Yin et al. (2025)	J	Zhou and Meng (2025)	J
Hazama and Iima (2025)	C	Verma et al. (2025)	C	Wang et al. (2025c)	C
Choi and An (2026)	J	Dong et al. (2026)	J	Ekinci et al. (2026)	J
Fang et al. (2026)	J	Song-Chol and Sung-Nam (2026)	J	Truong (2026)	J
Wen et al. (2026)	J	Wang (2026)	C	Song et al. (2026)	C
Sun and Cao (2026)	C				

Table B24: A collection of the references with the NLPSR mechanism. Publication type T is either J (journal paper) or C (conference paper or book-chapter).

Reference	T	Reference	T	Reference	T
Wong et al. (2017)	J	Bujok and Tvrdik (2017)	C	Awad et al. (2018)	J
Maucec et al. (2018)	J	Meng and Pan (2019)	J	Meng et al. (2019a)	J
Mohamed and Mohamed (2019)	C	Kumar et al. (2019)	C	Zhang et al. (2019)	C
Meng et al. (2020b)	J	Meng et al. (2020a)	J	Zhan et al. (2020)	J
Budiman et al. (2020)	C	Khan et al. (2020)	C	Zhang et al. (2020)	C
Meng and Yang (2021)	J	Meng et al. (2021)	J	Xia et al. (2021)	J
Zhong et al. (2021)	J	Stanovov et al. (2021)	C	Li et al. (2022b)	J
Meng and Yang (2022)	J	Najarro et al. (2022)	J	Pereira de Lacerda et al. (2022)	J
Yu et al. (2022)	J	Zeng et al. (2022)	J	Zheng and Luo (2022)	J
Biedrzycki et al. (2022)	C	Morales-Castaneda et al. (2022)	C	Ekinci et al. (2023)	J
Hong et al. (2023)	J	Li et al. (2023d)	J	Lv et al. (2023)	J
Meng (2023)	J	Meng and Zhang (2023b)	J	Sun et al. (2023a)	J
Tian et al. (2023)	J	Zhang et al. (2023a)	J	Zhang et al. (2023b)	J
Cao et al. (2024)	J	Liu et al. (2024)	J	Meng et al. (2024)	J
Qin et al. (2024)	J	Salgotra and Gandomi (2024)	J	Sedelnikov et al. (2024)	J
Yang et al. (2024b)	J	Zhang et al. (2024)	J	Zhou and Huang (2024)	J
Peng and Chen (2024)	C	Budiman et al. (2025)	J	Shen et al. (2025)	J
Sun et al. (2025)	J	Wu et al. (2025)	J	Xu et al. (2025)	J
Zeng et al. (2025)	J	Zhang et al. (2026)	J		

Appendix C Friedman test on CEC 2024 benchmark suite by function family

Figures C1 and C16 show Friedman mean ranks of the compared algorithms for different function families and problem dimensions. They correspond to Section 6.3.5.

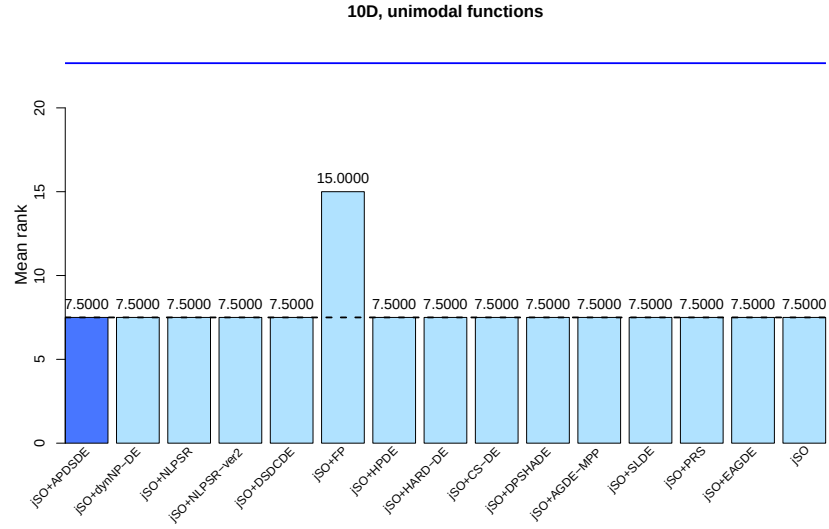


Fig. C1: Friedman ranks of algorithms in 10D (unimodal functions) with marked critical distance for jSO+APDSDE.

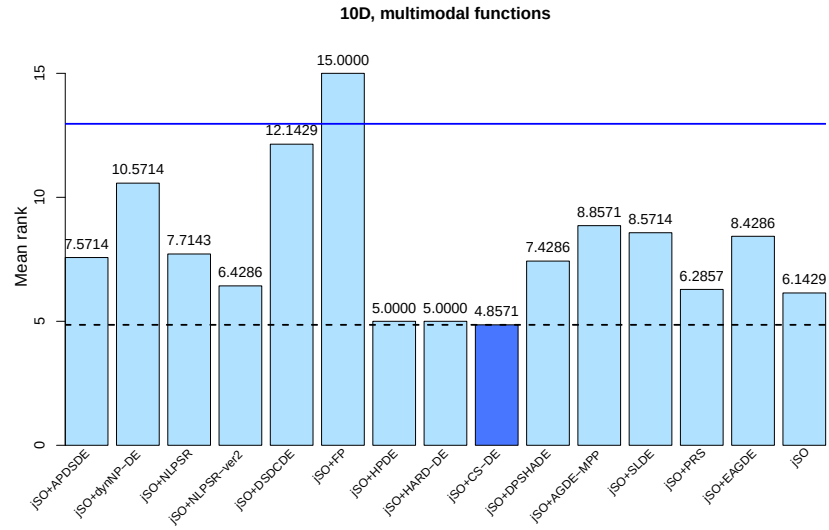


Fig. C2: Friedman ranks of algorithms in 10D (multimodal functions) with marked critical distance for jSO+CS-DE.

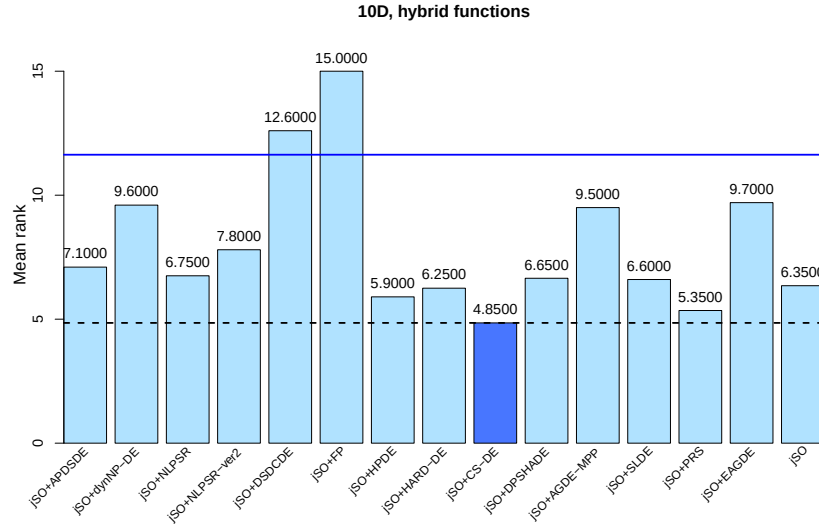


Fig. C3: Friedman ranks of algorithms in 10D (hybrid functions) with marked critical distance for jSO+CS-DE.

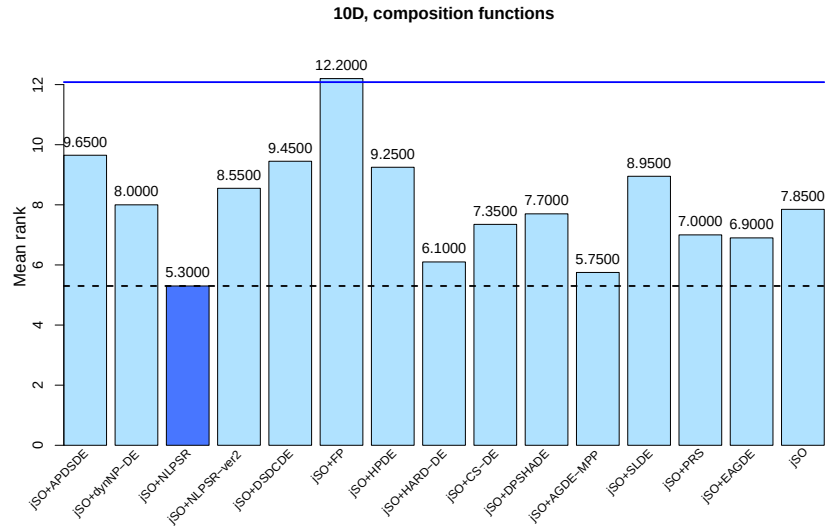


Fig. C4: Friedman ranks of algorithms in 10D (composition functions) with marked critical distance for jSO+NLPSR.

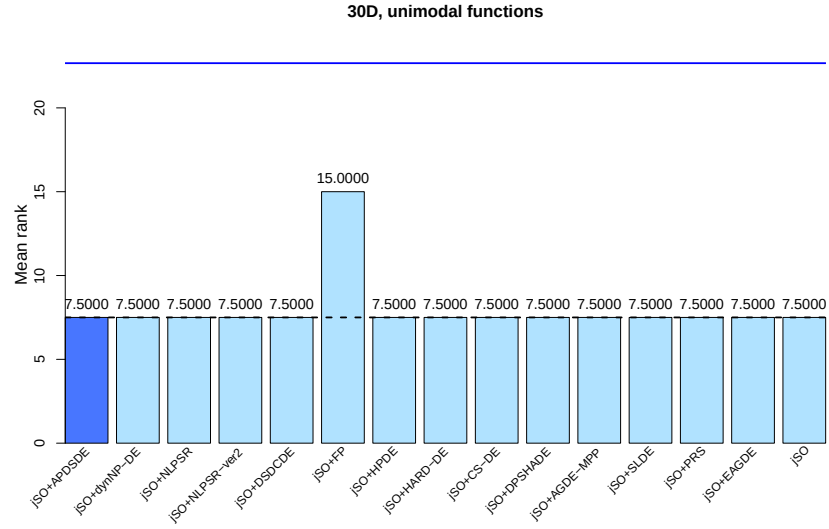


Fig. C5: Friedman ranks of algorithms in 30D (unimodal functions) with marked critical distance for jSO+APDSDE.

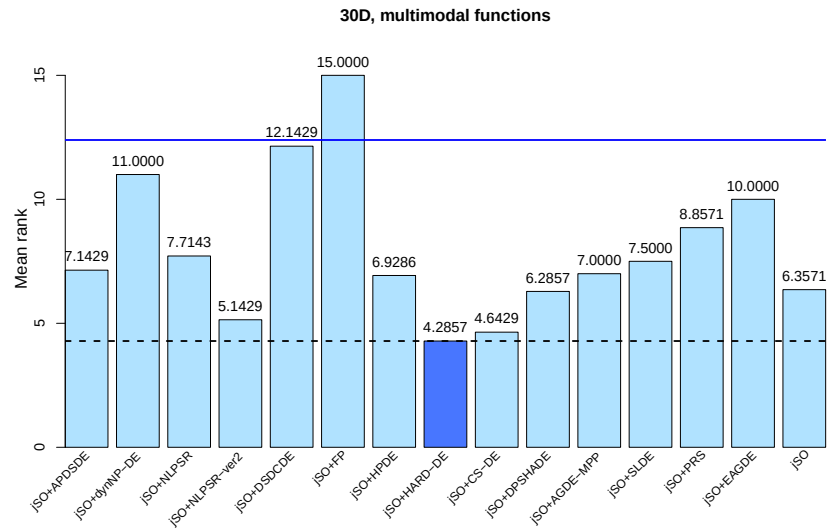


Fig. C6: Friedman ranks of algorithms in 30D (multimodal functions) with marked critical distance for jSO+HARD-DE.

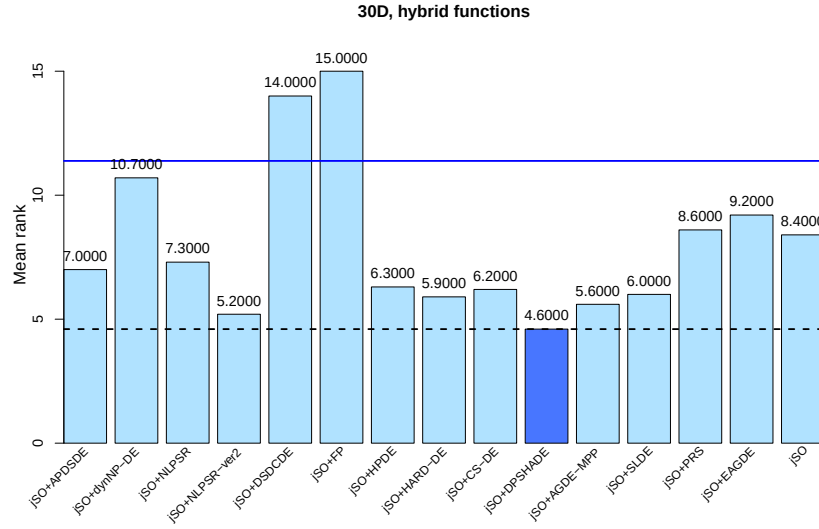


Fig. C7: Friedman ranks of algorithms in 30D (hybrid functions) with marked critical distance for jSO+DPSHADE.

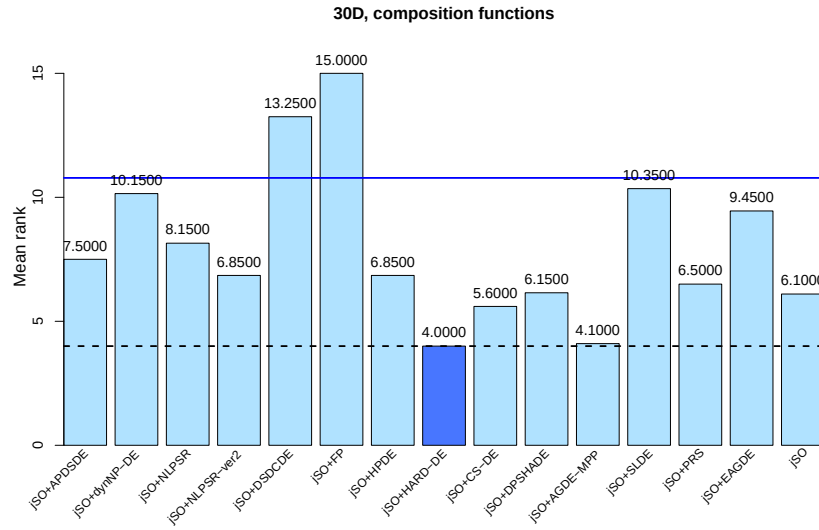


Fig. C8: Friedman ranks of algorithms in 30D (composition functions) with marked critical distance for jSO+HARD-DE.

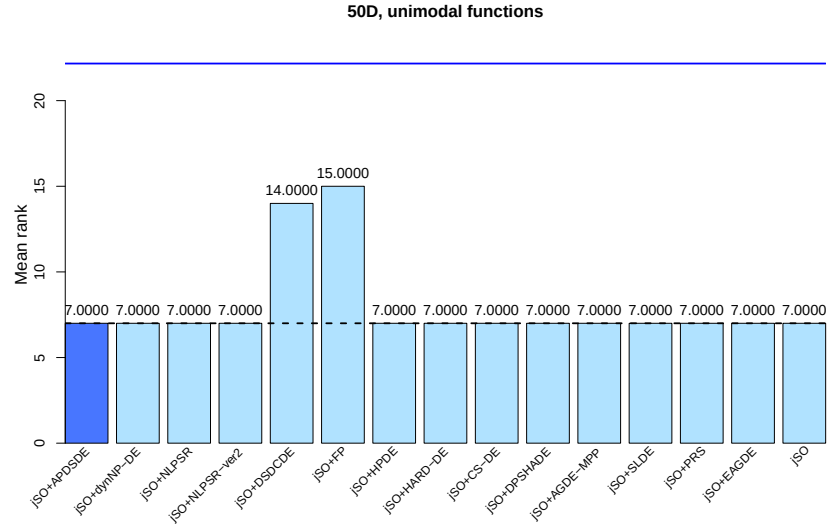


Fig. C9: Friedman ranks of algorithms in 50D (unimodal functions) with marked critical distance for jSO+APDSDE.

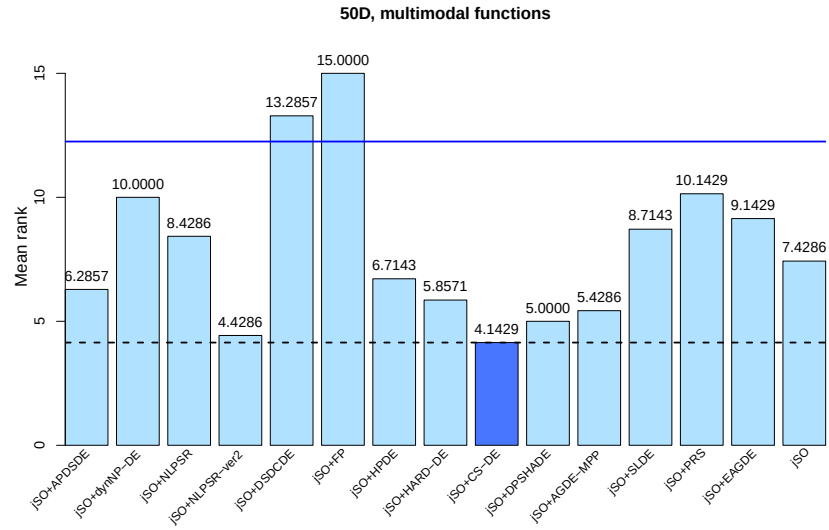


Fig. C10: Friedman ranks of algorithms in 50D (multimodal functions) with marked critical distance for jSO+CS-DE.

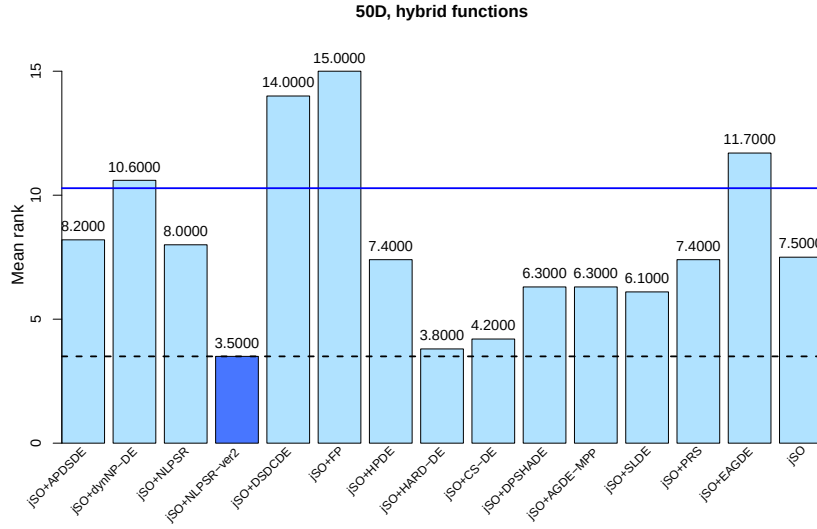


Fig. C11: Friedman ranks of algorithms in 50D (hybrid functions) with marked critical distance for jSO+NLPSR-ver2.

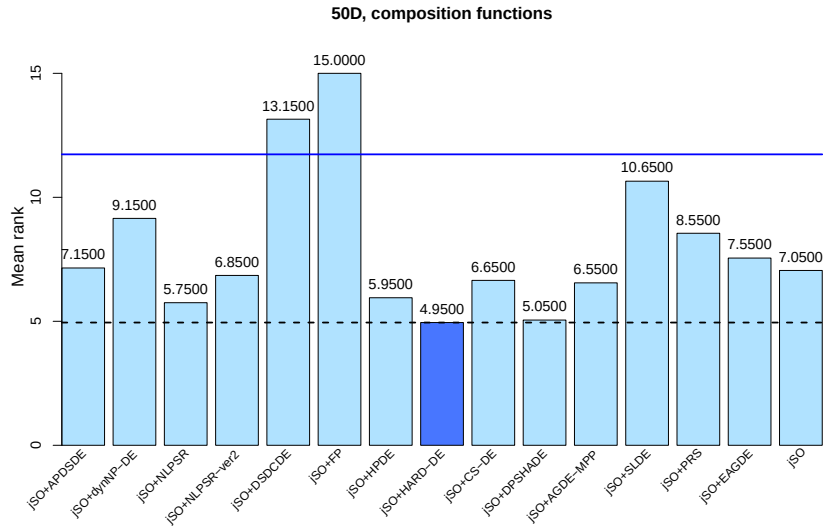


Fig. C12: Friedman ranks of algorithms in 50D (compositions functions) with marked critical distance for jSO+HARD-DE.

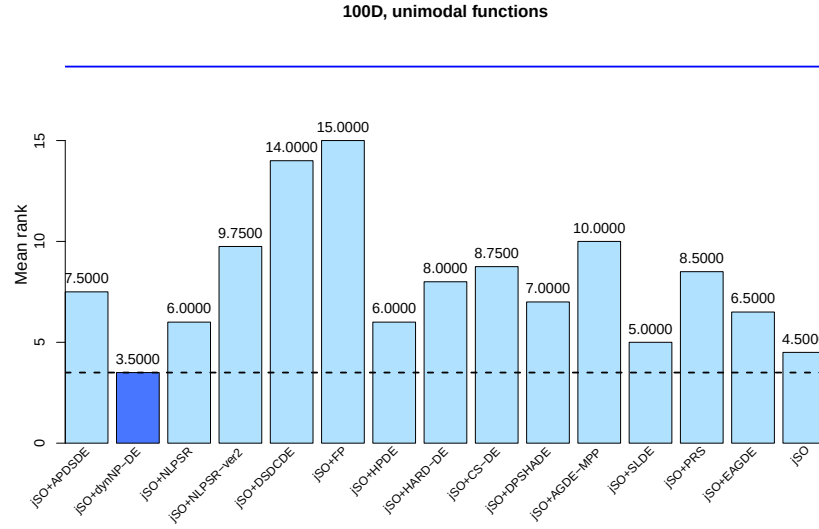


Fig. C13: Friedman ranks of algorithms in 100D (unimodal functions) with marked critical distance for jSO+dynNP-DE.

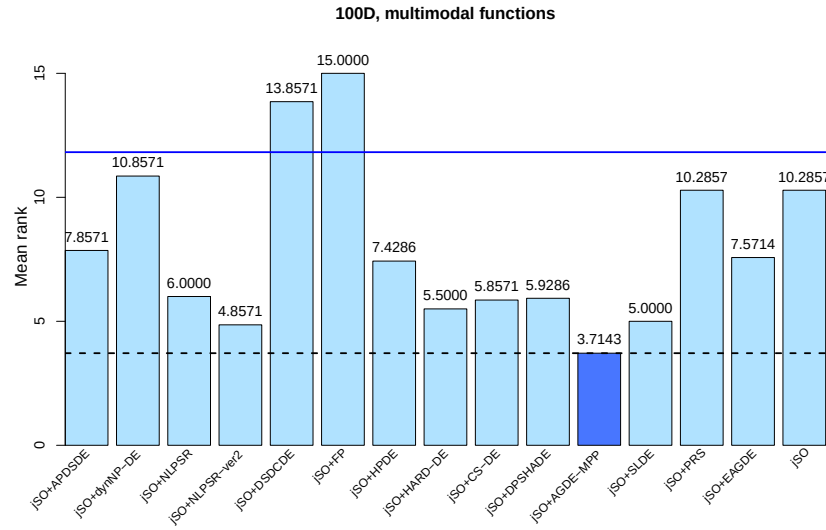


Fig. C14: Friedman ranks of algorithms in 100D (multimodal functions) with marked critical distance for jSO+AGDE-MPP.

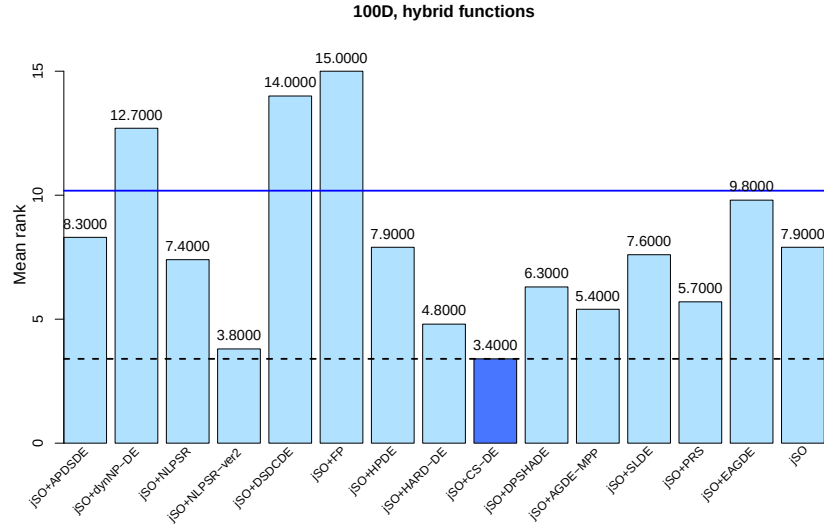


Fig. C15: Friedman ranks of algorithms in 100D (hybrid functions) with marked critical distance for jSO+CS-DE.

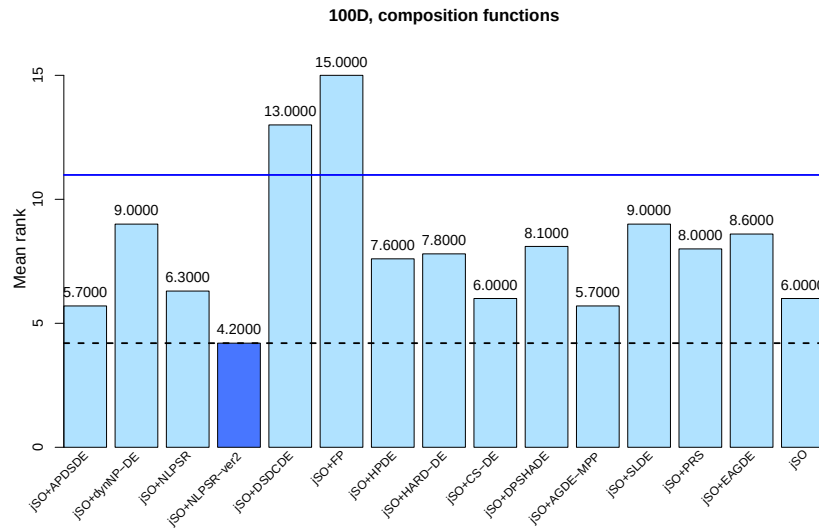


Fig. C16: Friedman ranks of algorithms in 100D (composition functions) with marked critical distance for jSO+NLPSR-ver2.

References

- Ahmed EM, Aly M, Mostafa M, et al (2023) An Accurate Model for Bifacial Photovoltaic Panels. *Sustainability (Switzerland)* 15(1). <https://doi.org/10.3390/su15010509>
- Al-Dabbagh RD, Neri F, Idris N, et al (2018) Algorithmic design issues in adaptive differential evolution schemes: Review and taxonomy. *Swarm and Evolutionary Computation* 43:284–311. <https://doi.org/10.1016/j.swevo.2018.03.008>
- Aly M, Mohamed EA, Rezk H, et al (2023) An Improved Optimally Designed Fuzzy Logic-Based MPPT Method for Maximizing Energy Extraction of PEMFC in Green Buildings. *Energies* 16(3). <https://doi.org/10.3390/en16031197>
- Aranguren I, Valdivia A, Masorales-Castañeda B, et al (2021) Improving the segmentation of magnetic resonance brain images using the LSHADE optimization algorithm. *Biomedical Signal Processing and Control* 64:102259. <https://doi.org/10.1016/j.bspc.2020.102259>
- Awad NH, Ali MZ, Liang JJ, et al (2016) Problem Definitions and Evaluation Criteria for the CEC 2017 Special Session and Competition on Single Objective Bound Constrained Real-Parameter Numerical Optimization. Tech. rep., Nanyang Technological University, Singapore, URL <https://github.com/P-N-Suganthan/CEC2017-BoundConstrained>
- Awad NH, Ali MZ, Suganthan PN (2017) Ensemble sinusoidal differential covariance matrix adaptation with Euclidean neighborhood for solving CEC2017 benchmark problems. In: 2017 IEEE congress on evolutionary computation (CEC), IEEE, pp 372–379, <https://doi.org/10.1109/CEC.2017.7969336>
- Awad NH, Ali MZ, Suganthan PN (2018) Ensemble of parameters in a sinusoidal differential evolution with niching-based population reduction. *Swarm and evolutionary computation* 39:141–156. <https://doi.org/10.1016/j.swevo.2017.09.009>
- Biedrzycki R, Arabas J, Jagodziński D (2019) Bound constraints handling in Differential Evolution: An experimental study. *Swarm and Evolutionary Computation* 50:100453. <https://doi.org/10.1016/j.swevo.2018.10.004>
- Biedrzycki R, Arabas J, Warchulski E (2022) A Version of NL-SHADE-RSP Algorithm with Midpoint for CEC 2022 Single Objective Bound Constrained Problems. In: 2022 IEEE Congress on Evolutionary Computation, CEC 2022 - Conference Proceedings, pp 1–8, <https://doi.org/10.1109/CEC55065.2022.9870220>
- Bilal, Pant M, Zaheer H, et al (2020) Differential Evolution: A review of more than two decades of research. *Engineering Applications of Artificial Intelligence* 90:103479. <https://doi.org/10.1016/j.engappai.2020.103479>

- Biswas P, Suganthan P, Amaratunga G (2018a) Optimal Power Flow Solutions Using Algorithm Success History Based Adaptive Differential Evolution with Linear Population Reduction. In: Proceedings - 2018 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2018, pp 249–254, <https://doi.org/10.1109/SMC.2018.00053>
- Biswas PP, Suganthan PN (2020) Large Initial Population and Neighborhood Search incorporated in LSHADE to solve CEC2020 Benchmark Problems. In: 2020 IEEE Congress on Evolutionary Computation (CEC), pp 1–7, <https://doi.org/10.1109/CEC48606.2020.9185547>
- Biswas PP, Awad NH, Suganthan P, et al (2017a) Minimizing THD of multilevel inverters with optimal values of DC voltages and switching angles using LSHADE-EpSin algorithm. In: 2017 IEEE Congress on Evolutionary Computation, CEC 2017 - Proceedings, pp 77–82, <https://doi.org/10.1109/CEC.2017.7969298>
- Biswas PP, Mallipeddi R, Suganthan P, et al (2017b) Optimal reconfiguration and distributed generator allocation in distribution network using an advanced adaptive differential evolution. In: 2017 IEEE Symposium Series on Computational Intelligence, SSCI 2017 - Proceedings, pp 1–7, <https://doi.org/10.1109/SSCI.2017.8280824>
- Biswas PP, Suganthan P, Amaratunga GA (2017c) Minimizing harmonic distortion in power system with optimal design of hybrid active power filter using differential evolution. *Applied Soft Computing Journal* 61:486–496. <https://doi.org/10.1016/j.asoc.2017.08.031>
- Biswas PP, Suganthan P, Amaratunga GAJ (2017d) Optimal placement of wind turbines in a windfarm using L-SHADE algorithm. In: 2017 IEEE Congress on Evolutionary Computation, CEC 2017 - Proceedings, pp 83–88, <https://doi.org/10.1109/CEC.2017.7969299>
- Biswas PP, Suganthan P, Amaratunga GA (2018b) Distribution Network Reconfiguration Together with Distributed Generator and Shunt Capacitor Allocation for Loss Minimization. In: 2018 IEEE Congress on Evolutionary Computation, CEC 2018 - Proceedings, pp 1–7, <https://doi.org/10.1109/CEC.2018.8477894>
- Biswas PP, Suganthan P, Wu G, et al (2019) Parameter estimation of solar cells using datasheet information with the application of an adaptive differential evolution algorithm. *Renewable Energy* 132:425–438. <https://doi.org/10.1016/j.renene.2018.07.152>
- Biswas S, Saha D, De S, et al (2021) Improving differential evolution through Bayesian hyperparameter optimization. In: 2021 IEEE Congress on evolutionary computation (CEC), IEEE, pp 832–840, <https://doi.org/10.1109/CEC45853.2021.9504792>

- Blaga AC, Dragoi EN, Cascaval D, et al (2025) Extraction of mandelic acid with ionic liquids: parametric study, model and process optimization with L-SHADE. *Scientific Reports* 15(1):42677. <https://doi.org/10.1038/s41598-025-26825-0>
- Brest J, Sepesy Maučec M (2008) Population size reduction for the differential evolution algorithm. *Applied Intelligence* 29(3):228–247. <https://doi.org/10.1007/s10489-007-0091-x>
- Brest J, Sepesy Maučec M (2025) Comparative study of modern differential evolution algorithms: Perspectives on mechanisms and performance. *Mathematics* 13(10). <https://doi.org/10.3390/math13101556>
- Brest J, Greiner S, Boskovic B, et al (2006) Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *IEEE transactions on evolutionary computation* 10(6):646–657. <https://doi.org/10.1109/TEVC.2006.872133>
- Brest J, Maučec MS, Bošković B (2016) iL-SHADE: Improved L-SHADE algorithm for single objective real-parameter optimization. In: 2016 IEEE Congress on Evolutionary Computation (CEC), pp 1188–1195, <https://doi.org/10.1109/CEC.2016.7743922>
- Brest J, Maučec MS, Bošković B (2017) Single objective real-parameter optimization: Algorithm jSO. In: 2017 IEEE congress on evolutionary computation (CEC), IEEE, Donostia, Spain, pp 1311–1318, <https://doi.org/10.1109/CEC.2017.7969456>
- Brest J, Maučec MS, Bošković B (2021) Self-adaptive Differential Evolution Algorithm with Population Size Reduction for Single Objective Bound-Constrained Optimization: Algorithm j21. In: 2021 IEEE Congress on Evolutionary Computation, CEC 2021 - Proceedings, pp 817–824, <https://doi.org/10.1109/CEC45853.2021.9504782>
- Budiman H, Li Wang S, Morsidi F, et al (2020) Self-Adaptive Ensemble-based Differential Evolution with Enhanced Population Sizing. In: 2020 2nd International Conference on Cybernetics and Intelligent System, ICORIS 2020, pp 1–6, <https://doi.org/10.1109/ICORIS50180.2020.9320767>
- Budiman H, Wang SL, Ng TF, et al (2025) A study of adaptive population sizing in a self-adaptive differential evolution. *Results in Control and Optimization* 20:100585. <https://doi.org/10.1016/j.rico.2025.100585>
- Bujok P (2020) Cooperative Model of Evolutionary Algorithms and Real-World Problems. *Communications in Computer and Information Science* 1092 CCIS:1–12. https://doi.org/10.1007/978-3-030-37838-7_1
- Bujok P (2021) The real-life application of differential evolution with a distance-based mutation-selection. *Mathematics* 9(16). <https://doi.org/10.3390/math9161909>

- Bujok P, Kolenovsky P (2022) Eigen Crossover in Cooperative Model of Evolutionary Algorithms Applied to CEC 2022 Single Objective Numerical Optimisation. In: 2022 IEEE Congress on Evolutionary Computation, CEC 2022 - Conference Proceedings, pp 1–8, <https://doi.org/10.1109/CEC55065.2022.9870433>
- Bujok P, Polaková R (2018) Migration Model of jSO Algorithm. In: 2018 25th International Conference on Systems, Signals and Image Processing (IWSSIP), pp 1–5, <https://doi.org/10.1109/IWSSIP.2018.8439569>
- Bujok P, Tvrdik J (2017) Enhanced individual-dependent differential evolution with population size adaptation. In: 2017 IEEE Congress on Evolutionary Computation, CEC 2017 - Proceedings, pp 1358–1365, <https://doi.org/10.1109/CEC.2017.7969462>
- Bujok P, Zamuda A (2019) Cooperative Model of Evolutionary Algorithms Applied to CEC 2019 Single Objective Numerical Optimization. In: 2019 IEEE Congress on Evolutionary Computation, CEC 2019 - Proceedings, pp 366–371, <https://doi.org/10.1109/CEC.2019.8790317>
- Cao Z, Xu K, Wang Z, et al (2024) An adaptive population size based Differential Evolution by mining historical population similarity for path planning of unmanned aerial vehicles. *Information Sciences* 666:120432. <https://doi.org/10.1016/j.ins.2024.120432>
- Chang Y, Wei X, Xu H, et al (2022) Study on Electromagnetic Scattering Characteristics of Inductively Coupled Plasma Superimposed Controllable Coding Diffuse Metasurface. *IEEE Transactions on Plasma Science* 50(12):4834–4842. <https://doi.org/10.1109/TPS.2022.3212372>
- Chen B, Ouyang H, Li S, et al (2025) Dual-stage self-adaptive differential evolution with complementary and ensemble mutation strategies. *Swarm and Evolutionary Computation* 93:101855. <https://doi.org/10.1016/j.swevo.2025.101855>
- Chen G, Shen Y, Qu N, et al (2021a) Path planning of AUV during diving process based on behavioral decision-making. *Ocean Engineering* 234:109073. <https://doi.org/10.1016/j.oceaneng.2021.109073>
- Chen G, Shen Y, Qu N, et al (2021b) Control architecture of autonomous underwater vehicle for coverage mission in irregular region. *Ocean Engineering* 236:109407. <https://doi.org/10.1016/j.oceaneng.2021.109407>
- Chen G, Shen Y, Zhang Y, et al (2021c) 2D multi-area coverage path planning using L-SHADE in simulated ocean survey. *Applied Soft Computing* 112:107754. <https://doi.org/10.1016/j.asoc.2021.107754>
- Cheng L, Wang Y, Wang C, et al (2020) Adaptive differential evolution based on successful experience information. *IEEE Access* 8:164611–164636. <https://doi.org/>

- Cheng L, Zhou JX, Hu X, et al (2024) Adaptive differential evolution with fitness-based crossover rate for global numerical optimization. *Complex and Intelligent Systems* 10(1):551–576. <https://doi.org/10.1007/s40747-023-01159-4>
- Cherifi A, Khanouche ME, Amirat Y, et al (2023) A parallel approach for user-centered QoS-aware services composition in the Internet of Things. *Engineering Applications of Artificial Intelligence* 123:106277. <https://doi.org/10.1016/j.engappai.2023.106277>
- Choi TJ, An Y (2026) Adaptive dimension-wise Cauchy perturbation for enhanced differential evolution optimization. *AIMS Mathematics* 11(1):734–766. <https://doi.org/10.3934/math.2026032>
- Choi TJ, Lee JH, Youn HY, et al (2019) Adaptive differential evolution with elite opposition-based learning and its application to training artificial neural networks. *Fundamenta Informaticae* 164(2-3):227–242. <https://doi.org/10.3233/FI-2019-1764>
- Cui Z, Zhao B, Zhao T, et al (2024) An adaptive differential evolution algorithm based on archive reuse. *Information Sciences* 668:120524. <https://doi.org/10.1016/j.ins.2024.120524>
- Das S, Suganthan PN (2010) Problem definitions and evaluation criteria for CEC 2011 competition on testing evolutionary algorithms on real world optimization problems. Jadavpur University, Nanyang Technological University, Kolkata pp 341–359. URL https://github.com/P-N-Suganthan/CEC-2011--Real_World_Problems
- Das S, Suganthan PN (2011) Differential evolution: A survey of the state-of-the-art. *IEEE transactions on evolutionary computation* 15(1):4–31. <https://doi.org/10.1109/TEVC.2010.2059031>
- Das S, Mullick SS, Suganthan PN (2016) Recent advances in differential evolution—an updated survey. *Swarm and evolutionary computation* 27:1–30. <https://doi.org/10.1016/j.swevo.2016.01.004>
- Deng L, Meng Z, Li C, et al (2025) Differential evolution with dual-space-based population size adaptation and dynamic classification-based mutation strategy. *Applied Intelligence* 55(10):750. <https://doi.org/10.1007/s10489-025-06609-9>
- Dinh TV, Nguyen H, Tran XL, et al (2021) Predicting Rainfall-Induced Soil Erosion Based on a Hybridization of Adaptive Differential Evolution and Support Vector Machine Classification. *Mathematical Problems in Engineering* 2021(1):6647829. <https://doi.org/10.1155/2021/6647829>
- Dong C, Li Z, Li Y, et al (2026) Error calibration for cross-shaped magnetic gradient tensor system based on the improved differential evolution algorithm. *Journal of*

- Applied Geophysics 245:106080. <https://doi.org/10.1016/j.jappgeo.2025.106080>
- Dong S, Yang Y, Xu W, et al (2023) L-CJADE: Differential Evolution Algorithm Based On A Linearly Decreasing Population Structure. In: 2023 8th International Conference on Computer and Communication Systems (ICCCS), pp 875–879, <https://doi.org/10.1109/ICCCS57501.2023.10151181>
- Dragoi EN, Dafinescu V (2016) Parameter control and hybridization techniques in differential evolution: a survey. Artificial Intelligence Review 45(4):447–470. <https://doi.org/10.1007/s10462-015-9452-8>
- Durmuş B, Ünlü A, Özyön S, et al (2025) Optimal Filter Design Using Differential Evolution Algorithm and Its Advanced Variants. Circuits, Systems, and Signal Processing 44(10):7166–7205. <https://doi.org/10.1007/s00034-025-03146-1>
- Ekinci S, Izci D, Almeree A, et al (2026) Temperature Control of Nonlinear Continuous Stirred Tank Reactors Using an Enhanced Nature-Inspired Optimizer and Fractional-Order Controller. Biomimetics 11(2). <https://doi.org/10.3390/biomimetics11020153>
- Ekinci YL, Balkaya Ç, Göktürkler G, et al (2023) 3-D gravity inversion for the basement relief reconstruction through modified success-history-based adaptive differential evolution. Geophysical Journal International 235(1):377–400. <https://doi.org/10.1093/gji/ggad222>
- El-Abd M, Yu K, Ge S (2020) Parameters identification of photovoltaic cell and module using LSHADE. In: 12th International Conference on Advanced Computational Intelligence, ICACI 2020, pp 189–193, <https://doi.org/10.1109/ICACI49185.2020.9177843>
- Emami S, Emami H, Parsa J (2023) LXGB: a machine learning algorithm for estimating the discharge coefficient of pseudo-cosine labyrinth weir. Scientific Reports 13(1):12304. <https://doi.org/10.1038/s41598-023-39272-6>
- Fang Y, Qu W, Xie Z, et al (2026) Experimental study of NH₃/H₂/n-heptane combustion and reduced mechanism optimization via a CNN-augmented neural network and the L-SHADE algorithm. Combustion and Flame 286:114871. <https://doi.org/10.1016/j.combustflame.2026.114871>
- Fu S, Ma C, Li K, et al (2025) Modified LSHADE-SPACMA with new mutation strategy and external archive mechanism for numerical optimization and point cloud registration. Artificial Intelligence Review 58(3):72. <https://doi.org/10.1007/s10462-024-11053-1>
- Gehlot N, Vijayvargiya A, Jena A, et al (2025) L-SHADE optimized learning framework for sEMG hand gesture recognition. Scientific Reports 15(1):36562. <https://doi.org/10.1038/s41598-025-20076-9>

- Ghosh A, Das S, Das AK, et al (2022) Using spatial neighborhoods for parameter adaptation: An improved success history based differential evolution. *Swarm and Evolutionary Computation* 71:101057. <https://doi.org/10.1016/j.swevo.2022.101057>
- Gong J, Laili Y, Zhang J, et al (2025) A triple population adaptive differential evolution. *Information Sciences* 688:121401. <https://doi.org/10.1016/j.ins.2024.121401>
- Goudos SK, Tsoulos GV, Athanasiadou G, et al (2019) Artificial Neural Network Optimal Modeling and Optimization of UAV Measurements for Mobile Communications Using the L-SHADE Algorithm. *IEEE Transactions on Antennas and Propagation* 67(6):4022–4031. <https://doi.org/10.1109/TAP.2019.2905665>
- Gu Q, Li S, Gong W, et al (2023a) L-SHADE with parameter decomposition for photovoltaic modules parameter identification under different temperature and irradiance. *Applied Soft Computing* 143:110386. <https://doi.org/10.1016/j.asoc.2023.110386>
- Gu Z, Xiong G, Fu X, et al (2023b) Extracting accurate parameters of photovoltaic cell models via elite learning adaptive differential evolution. *Energy Conversion and Management* 285:116994. <https://doi.org/10.1016/j.enconman.2023.116994>
- Gupta S, Singh S, Gao S (2025) Fitness and Collaborative Information-Driven Differential Evolution Algorithm for Bound-Constrained and Real-World Application Problems. *Arabian Journal for Science and Engineering* 50(19):16017–16051. <https://doi.org/10.1007/s13369-025-10081-5>
- Hadi AA, Mohamed AW, Jambi KM (2019) LSHADE-SPA memetic framework for solving large-scale optimization problems. *Complex and Intelligent Systems* 5(1):25–40. <https://doi.org/10.1007/s40747-018-0086-8>
- Hadi AA, Mohamed AW, Jambi KM (2021) Single-objective real-parameter optimization: Enhanced LSHADE-SPACMA algorithm. In: *Heuristics for optimization and learning*. Springer, p 103–121, https://doi.org/10.1007/978-3-030-58930-1_7
- Hamdi M, Idoumghar L, Chaoui M, et al (2018) Optimal economic dispatch with valve-point effect and multiple fuel option via L-shade algorithm. In: *2018 15th International Multi-Conference on Systems, Signals and Devices, SSD 2018*, pp 537–542, <https://doi.org/10.1109/SSD.2018.8570673>
- Hamdi M, Idomghar L, Chaoui M, et al (2019) An improved adaptive differential evolution optimizer for non-convex Economic Dispatch Problems. *Applied Soft Computing Journal* 85:105868. <https://doi.org/10.1016/j.asoc.2019.105868>
- Hao Q, Zhou Z, Wei Z, et al (2020) Parameters Identification of Photovoltaic Models Using a Multi-Strategy Success-History-Based Adaptive Differential Evolution.

- IEEE Access 8:35979–35994. <https://doi.org/10.1109/ACCESS.2020.2975078>
- Hazama Y, Iima H (2025) Memory Switching L-Shade Based on Efficient Non-Convex Ratio. In: 2025 SICE Festival with Annual Conference, SICE FES 2025, pp 1029–1033, <https://doi.org/10.23919/SICEFES67750.2025.11236570>
- Hoang ND (2020) Image Processing-Based Pitting Corrosion Detection Using Metaheuristic Optimized Multilevel Image Thresholding and Machine-Learning Approaches. *Mathematical Problems in Engineering* 2020:6765274. <https://doi.org/10.1155/2020/6765274>
- Hoang ND, Nguyen QL (2020) A Novel Approach for Automatic Detection of Concrete Surface Voids Using Image Texture Analysis and History-Based Adaptive Differential Evolution Optimized Support Vector Machine. *Advances in Civil Engineering* 2020:4190682. <https://doi.org/10.1155/2020/4190682>
- Hong L, Guo Y, Liu F, et al (2023) A variant of the united multi-operator evolutionary algorithms using sequential quadratic programming and improved SHADE-cnEpSin. *Information Sciences* 622:652–681. <https://doi.org/10.1016/j.ins.2022.11.131>
- Hraiz R, Khader M, Ali MZ, et al (2017) Evaluating the Performance of "Derandomized-LShade" Algorithm on CEC 2014 Benchmark Functions. In: 2017 International Conference on New Trends in Computing Sciences (ICTCS), pp 36–41, <https://doi.org/10.1109/ICTCS.2017.45>
- Huang Pq, Wang Y, Wang Kz (2020) Energy-efficient trajectory planning for a multi-UAV-assisted mobile edge computing system. *Frontiers of Information Technology and Electronic Engineering* 21(12):1713–1725. <https://doi.org/10.1631/FITEE.2000315>
- Huang Y, Xing L, Li B, et al (2024) PC-SSRDE: A paradigm crossover-based differential evolution algorithm with search space reduction. *Information Sciences* 681:121188. <https://doi.org/10.1016/j.ins.2024.121188>
- Huang YC, Chang WC, Hsu H, et al (2021) Planning and research of distribution feeder automation with decentralized power supply. *Electronics (Switzerland)* 10(3):1–18. <https://doi.org/10.3390/electronics10030362>
- Ji JY, Zeng S, Wong ML (2022) ε -Constrained multiobjective differential evolution using linear population size expansion. *Information Sciences* 609:445–464. <https://doi.org/10.1016/j.ins.2022.07.108>
- Jiang Y, Zhan ZH, Tan KC, et al (2025) Knowledge learning for evolutionary computation. *IEEE transactions on evolutionary computation* 29(1):16–30. <https://doi.org/10.1109/TEVC.2023.3278132>

- Khan KH, Thapa KB, Karki NR (2020) Optimal coordination of directional over-current relays using enhanced L-SHADE Algorithm. In: 2020 IEEE International Conference on Power Systems Technology (POWERCON), IEEE, pp 1–6, <https://doi.org/10.1109/POWERCON48463.2020.9230593>
- Khodkari N, Hamidian P, Khodkari H, et al (2024) Predicting the small strain shear modulus of sands and sand-fines binary mixtures using machine learning algorithms. *Transportation Geotechnics* 44:101172. <https://doi.org/10.1016/j.trgeo.2023.101172>
- Kitamura T, Fukunaga A (2022) Differential evolution with an unbounded population. In: 2022 IEEE Congress on Evolutionary Computation (CEC), IEEE, pp 1–8, <https://doi.org/10.1109/CEC55065.2022.9870363>
- Kostovska A, Vermetten D, Korošec P, et al (2025) Using Machine Learning Methods to Assess Module Performance Contribution in Modular Optimization Frameworks. *Evolutionary computation* 33(4):485–512. https://doi.org/10.1162/evco_a_00356
- Kou Z, Yang Y, Li Z, et al (2025) A monthly runoff prediction model based on ICEEMD-L-SHADE-SRU. *Journal of Freshwater Ecology* 40(1). <https://doi.org/10.1080/02705060.2025.2488017>
- Koçer HG, Uymaz SA (2021) A novel local search method for LSGO with golden ratio and dynamic search step. *Soft Computing* 25(3):2115–2130. <https://doi.org/10.1007/s00500-020-05284-x>
- Kumar A, Misra RK, Singh D, et al (2019) Testing A Multi-Operator based Differential Evolution Algorithm on the 100-Digit Challenge for Single Objective Numerical Optimization. In: 2019 IEEE Congress on Evolutionary Computation, CEC 2019 - Proceedings, pp 34–40, <https://doi.org/10.1109/CEC.2019.8789907>
- Pereira de Lacerda MG, Amorim Neto HdA, Ludermir TB, et al (2022) Towards a parameterless out-of-the-box population size control for evolutionary and swarm-based algorithms for single objective bound constrained real-parameter numerical optimization. *Applied Soft Computing* 123:101294. <https://doi.org/10.1016/j.asoc.2022.108920>
- Laredo JLJ, Fernandes C, Merelo JJ, et al (2009) Improving genetic algorithms performance via deterministic population shrinkage. In: Proceedings of the 11th Annual conference on Genetic and evolutionary computation, pp 819–826, <https://doi.org/10.1145/1569901.1570014>
- Lei X, Shangqin T, Zhenglei W, et al (2023) UCAV situation assessment method based on C-LSHADE-Means and SAE-LVQ. *Journal of Systems Engineering and Electronics* 34(5):1235–1251. <https://doi.org/10.23919/JSEE.2023.000062>

- Li G, Li Q, Wang Z (2022a) Adaptive Differential Evolution Algorithm with Multiple Gaussian Learning Models. *Lecture Notes in Computer Science* (including subseries *Lecture Notes in Artificial Intelligence* and *Lecture Notes in Bioinformatics*) 13606 LNAI:325–336. https://doi.org/10.1007/978-3-031-20503-3_26
- Li K, Fu X, Wang F, et al (2022b) A dynamic population reduction differential evolution algorithm combining linear and nonlinear strategy piecewise functions. *Concurrency and Computation: Practice and Experience* 34(6):e6773. <https://doi.org/10.1002/cpe.6773>
- Li Y, Han T, Zhou H, et al (2022c) A novel adaptive L-SHADE algorithm and its application in UAV swarm resource configuration problem. *Information Sciences* 606:350–367. <https://doi.org/10.1016/j.ins.2022.05.058>
- Li Y, Han T, Tang S, et al (2023a) An improved differential evolution by hybridizing with estimation-of-distribution algorithm. *Information Sciences* 619:439–456. <https://doi.org/10.1016/j.ins.2022.11.029>
- Li Y, Han T, Wang X, et al (2023b) MjSO: A modified differential evolution with a probability selection mechanism and a directed mutation strategy. *Swarm and Evolutionary Computation* 78:101294. <https://doi.org/10.1016/j.swevo.2023.101294>
- Li Y, Han T, Zhou H, et al (2023c) APSM-jSO: A novel jSO variant with an adaptive parameter selection mechanism and a new external archive updating mechanism. *Swarm and Evolutionary Computation* 78:101283. <https://doi.org/10.1016/j.swevo.2023.101283>
- Li Y, Wang S, Yang B, et al (2023d) Population reduction with individual similarity for differential evolution. *Artificial Intelligence Review* 56:3887–3949. <https://doi.org/10.1007/s10462-022-10264-8>
- Lin Y, Du W, Stutzle T (2017) Three L-SHADE based algorithms on mixed-variables optimization problems. In: *2017 IEEE Congress on Evolutionary Computation, CEC 2017 - Proceedings*, pp 2274–2281, <https://doi.org/10.1109/CEC.2017.7969580>
- Liu X, Sun J, Zhang Q, et al (2023) Learning to learn evolutionary algorithm: A learnable differential evolution. *IEEE Transactions on Emerging Topics in Computational Intelligence* 7(6):1605–1620. <https://doi.org/10.1109/TETCI.2023.3251441>
- Liu Y, Hu L, Ma Z (2024) A New Adaptive Differential Evolution Algorithm Fused with Multiple Strategies for Robot Path Planning. *Arabian Journal for Science and Engineering* 49(9):11907–11924. <https://doi.org/10.1007/s13369-023-08380-w>

- Liu ZG (2019) Linear Distribution-based SHADE with Variable Population Size. In: Proceedings of 2019 IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference, IAEAC 2019, pp 745–748, <https://doi.org/10.1109/IAEAC47372.2019.8997762>
- Lv D, Xiong G, Fu X, et al (2023) Exponential hybrid mutation differential evolution for economic dispatch of large-scale power systems considering valve-point effects. *Applied Intelligence* 53(24):31046–31064. <https://doi.org/10.1007/s10489-023-05180-5>
- Ma X, Wang J, Hou C, et al (2025) Reconstruction of EMI Sources for PCBs Based on Adaptive Differential Evolution Algorithm. *IEEE Transactions on Electromagnetic Compatibility* 67(6):1796–1805. <https://doi.org/10.1109/TEMPC.2025.3605016>
- Maucec MS, Brest J, Boskovic B, et al (2018) Improved Differential Evolution for Large-Scale Black-Box Optimization. *IEEE Access* 6:29516–29531. <https://doi.org/10.1109/ACCESS.2018.2842114>
- Meng Z (2023) Dimension improvements based adaptation of control parameters in Differential Evolution: A fitness-value-independent approach. *Expert Systems with Applications* 223:119848. <https://doi.org/10.1016/j.eswa.2023.119848>
- Meng Z, Pan JS (2019) HARD-DE: Hierarchical archive based mutation strategy with depth information of evolution for the enhancement of differential evolution on numerical optimization. *IEEE Access* 7:12832–12854. <https://doi.org/10.1109/ACCESS.2019.2893292>
- Meng Z, Yang C (2021) Hip-DE: Historical population based mutation strategy in differential evolution with parameter adaptive mechanism. *Information Sciences* 562:44–77. <https://doi.org/10.1016/j.ins.2021.01.031>
- Meng Z, Yang C (2022) Two-stage differential evolution with novel parameter control. *Information Sciences* 596:321–342. <https://doi.org/10.1016/j.ins.2022.03.043>
- Meng Z, Zhang J (2023a) QUATRE-EMS: QUATRE algorithm with novel adaptation of evolution matrix and selection operation for numerical optimization. *Information Sciences* 651:119714. <https://doi.org/10.1016/j.ins.2023.119714>
- Meng Z, Zhang Q (2023b) HPDE: A dynamic hierarchical population based differential evolution with novel diversity metric. *Engineering Applications of Artificial Intelligence* 126:106989. <https://doi.org/10.1016/j.engappai.2023.106989>
- Meng Z, Pan JS, Tseng KK (2019a) PaDE: An enhanced differential evolution algorithm with novel control parameter adaptation schemes for numerical optimization. *Knowledge-Based Systems* 168:80–99. <https://doi.org/10.1016/j.knosys.2019.01.006>

- Meng Z, Pan JS, Zheng WM, et al (2019b) An improved linear population size reduction based parameters with adaptive learning mechanism differential evolution (iLPALMDE) for real-parameter single objective black box optimization. *Smart Innovation, Systems and Technologies* 129:477–484. https://doi.org/10.1007/978-3-030-04582-1_56
- Meng Z, Chen Y, Li X (2020a) Enhancing Differential Evolution with Novel Parameter Control. *IEEE Access* 8:51145–51167. <https://doi.org/10.1109/ACCESS.2020.2979738>
- Meng Z, Chen Y, Li X, et al (2020b) PaDE-NPC: Parameter Adaptive Differential Evolution with Novel Parameter Control for Single-Objective Optimization. *IEEE Access* 8:139460–139478. <https://doi.org/10.1109/ACCESS.2020.3012885>
- Meng Z, Zhong Y, Yang C (2021) CS-DE: Cooperative strategy based differential evolution with population diversity enhancement. *Information Sciences* 577:663–696. <https://doi.org/10.1016/j.ins.2021.07.080>
- Meng Z, Lin X, Chen D (2024) ACD-DE: An adaptive cluster division Differential Evolution for mitigating population diversity deficiency. *Information Sciences* 679:121091. <https://doi.org/10.1016/j.ins.2024.121091>
- Mohamed AK, Mohamed AW (2019) Real-parameter unconstrained optimization based on enhanced AGDE algorithm. In: Hassanien AE (ed) *Machine learning paradigms: Theory and application*. Springer International Publishing, Cham, p 431–450, https://doi.org/10.1007/978-3-030-02357-7_21
- Mohamed AW, Hadi AA, Fattouh AM, et al (2017) LSHADE with semi-parameter adaptation hybrid with CMA-ES for solving CEC 2017 benchmark problems. In: 2017 IEEE Congress on Evolutionary Computation (CEC), pp 145–152, <https://doi.org/10.1109/CEC.2017.7969307>
- Morales-Castaneda B, MacIel-Castillo O, Navarro MA, et al (2022) Handling stagnation through diversity analysis: A new set of operators for evolutionary algorithms. In: 2022 IEEE Congress on Evolutionary Computation, CEC 2022 - Conference Proceedings, pp 1–7, <https://doi.org/10.1109/CEC55065.2022.9870284>
- Mousavirad SJ, Rahnamayan S (2020) Enhancing SHADE and L-SHADE Algorithms Using Ordered Mutation. In: 2020 IEEE Symposium Series on Computational Intelligence, SSCI 2020, pp 337–344, <https://doi.org/10.1109/SSCI47803.2020.9308550>
- Mousavirad SJ, Moghadam MH, Saadatmand M, et al (2022) RWS-L-SHADE: An Effective L-SHADE Algorithm Incorporation Roulette Wheel Selection Strategy for Numerical Optimisation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 13224 LNCS:255–268. https://doi.org/10.1007/978-3-031-02462-7_17

- Naas B, Nezli L, Elbar M, et al (2025) Speed Estimation of a Double-Star Permanent Magnet Synchronous Motor Using an Optimized Extended Kalman Filter. *International Transactions on Electrical Energy Systems* 2025(1):8466428. <https://doi.org/10.1155/etep/8466428>
- Najarro LAC, Song I, Kim K (2022) Adaptation of population size in differential evolution and its effects on localization of target nodes. *IEEE Access* 10:107785–107798. <https://doi.org/10.1109/ACCESS.2022.3213060>
- Nasir Y, Yu W, Sepehrnoori K (2020) Hybrid derivative-free technique and effective machine learning surrogate for nonlinear constrained well placement and production optimization. *Journal of Petroleum Science and Engineering* 186:106726. <https://doi.org/10.1016/j.petrol.2019.106726>
- Neri F, Tirronen V (2010) Recent advances in differential evolution: a survey and experimental analysis. *Artificial Intelligence Review* 33(1):61–106. <https://doi.org/10.1007/s10462-009-9137-2>
- Nguyen TD, Tran TH, Nguyen H, et al (2021) A success history-based adaptive differential evolution optimized support vector regression for estimating plastic viscosity of fresh concrete. *Engineering with Computers* 37(2):1485–1498. <https://doi.org/10.1007/s00366-019-00899-7>
- Ozkaya B, Guvenc U, Bingol O (2022) Fitness Distance Balance Based LSHADE Algorithm for Energy Hub Economic Dispatch Problem. *IEEE Access* 10:66770–66796. <https://doi.org/10.1109/ACCESS.2022.3185068>
- Page MJ, McKenzie JE, Bossuyt PM, et al (2021) The prisma 2020 statement: an updated guideline for reporting systematic reviews. *bmj* 372. <https://doi.org/10.1136/bmj.n71>
- Pan JS, Chen J, Li J, et al (2026) Differential evolution with exponential crossover: A survey. *Computer Science Review* 62:100990. <https://doi.org/10.1016/j.cosrev.2026.100990>
- Peng F, Chen X (2024) An Efficient Antenna Optimization Method Based on Variable Population Size. In: 2024 IEEE 12th Asia-Pacific Conference on Antennas and Propagation, APCAP 2024 - Proceedings, <https://doi.org/10.1109/APCAP62011.2024.10881078>
- Pholdee N, Bureerat S (2018) A Comparative Study of Eighteen Self-adaptive Metaheuristic Algorithms for Truss Sizing Optimisation. *KSCE Journal of Civil Engineering* 22(8):2982–2993. <https://doi.org/10.1007/s12205-017-0095-y>
- Piotrowska AE, Piotrowski AP (2026) Improving scale parameters in Successful-history-based adaptive Differential Evolution algorithms. *Applied Soft Computing* 187:114288. <https://doi.org/10.1016/j.asoc.2025.114288>

- Piotrowski AP (2017) Review of differential evolution population size. *Swarm and Evolutionary Computation* 32:1–24. <https://doi.org/10.1016/j.swevo.2016.05.003>
- Piotrowski AP (2018) L-SHADE optimization algorithms with population-wide inertia. *Information Sciences* 468:117–141. <https://doi.org/10.1016/j.ins.2018.08.030>
- Piotrowski AP, Napiorkowski MJ, Napiorkowski JJ, et al (2017) Swarm intelligence and evolutionary algorithms: Performance versus speed. *Information Sciences* 384:34–85. <https://doi.org/10.1016/j.ins.2016.12.028>
- Piotrowski AP, Piotrowska AE, Napiorkowski JJ (2026) Experimental survey of L-SHADE and SHADE-based adaptive differential evolution algorithms. *Swarm and Evolutionary Computation* 101:102286. <https://doi.org/10.1016/j.swevo.2026.102286>
- Poláková R, Tvrdík J, Bujok P (2019) Differential evolution with adaptive mechanism of population size according to current population diversity. *Swarm and Evolutionary Computation* 50:100519. <https://doi.org/10.1016/j.swevo.2019.03.014>
- Poláková R, Bujok P (2021) Popular Optimisation Algorithms with Diversity-Based Adaptive Mechanism for Population Size. *Studies in Fuzziness and Soft Computing* 403:171–182. https://doi.org/10.1007/978-3-030-61659-5_15
- Popič J, Bošković B, Brest J (2025) Improving mutation strategies in differential evolution with a new pbest selection mechanism. *Applied Soft Computing* 185:113978. <https://doi.org/10.1016/j.asoc.2025.113978>
- Price KV, Storn RM, Lampinen JA (2005) *Differential evolution: a practical approach to global optimization*. Springer
- Qiao K, Wen X, Ban X, et al (2023) Evaluation criteria for CEC 2024 competition and special session on numerical optimization considering accuracy and speed. Zhengzhou University, Central South University, Henan Institute of Technology, Qatar University, Tech Rep URL https://github.com/P-N-Suganthan/2024-CEC/blob/main/Tech_Report_U_Score_Comparison.pdf
- Qin Y, Deng L, Li C, et al (2024) A dimensional difference-based population size adjustment framework for differential evolution. *Information Sciences* 660:120110. <https://doi.org/10.1016/j.ins.2024.120110>
- Refaat MM, Aleem SHEA, Atia Y, et al (2021) Multi-stage dynamic transmission network expansion planning using lshade-spacma. *Applied Sciences (Switzerland)* 11(5):1–22. <https://doi.org/10.3390/app11052155>
- Reyes-Davila E, Haro EH, Casas-Ordaz A, et al (2025) Differential evolution: a survey on their operators and variants. *Archives of Computational Methods in Engineering* 32(1):83–112. <https://doi.org/10.1007/s11831-024-10136-0>

- de Rezende Freitas EJ, Cohen MW, Guimarães FG (2023) An Autonomous Mobile Robot Path Planner using Spline Curves and Differential Evolution. In: 2023 IEEE Latin American Conference on Computational Intelligence, LA-CCI 2023, pp 1–6, <https://doi.org/10.1109/LA-CCI58595.2023.10409335>
- Rizk-Allah RM, El-Fergany AA (2021) Effective coordination settings for directional overcurrent relay using hybrid Gradient-based optimizer. *Applied Soft Computing* 112:107748. <https://doi.org/10.1016/j.asoc.2021.107748>
- Saad A, Engelbrecht AP, Khan SA (2024) An analysis of differential evolution population size. *Applied Sciences* 14(21):9976. <https://doi.org/10.3390/app14219976>
- Salgotra R, Gandomi AH (2024) A novel multi-hybrid differential evolution algorithm for optimization of frame structures. *Scientific Reports* 14(1):4877. <https://doi.org/10.1038/s41598-024-54384-3>
- Salgotra R, Singh U, Saha S, et al (2019) New Improved SALSHADE-cnEpSin Algorithm with Adaptive Parameters. In: 2019 IEEE Congress on Evolutionary Computation, CEC 2019 - Proceedings, pp 3150–3156, <https://doi.org/10.1109/CEC.2019.8789983>
- Salgotra R, Singh S, Singh U, et al (2022) An Adaptive Version of Differential Evolution for Solving CEC2014, CEC 2017 and CEC 2022 Test Suites. In: Proceedings of the 2022 IEEE Symposium Series on Computational Intelligence, SSCI 2022, pp 1644–1649, <https://doi.org/10.1109/SSCI51031.2022.10022308>
- Sallam KM, Chakraborty RK, Ryan MJ (2019) A Hybrid Differential Evolution with Cuckoo Search for Solving Resource Constrained Project Scheduling Problems. In: IEEE International Conference on Industrial Engineering and Engineering Management, pp 1344–1348, <https://doi.org/10.1109/IEEM44572.2019.8978856>
- Sallam KM, Alrashdi I, Mohamed R, et al (2025) An enhanced LSHADE-based algorithm for global and constrained optimization in applied mechanics and power flow problems. *Swarm and Evolutionary Computation* 97:102032. <https://doi.org/10.1016/j.swevo.2025.102032>
- Sarajlić M, Hribernik M, Kitak P, et al (2019) Identification of the Thermal Parameters for the NH gG Fuse Using the Differential Evolution. *IEEE Access* 7:55640–55648. <https://doi.org/10.1109/ACCESS.2019.2913553>
- Sato J, Akashi T, Yamada T, et al (2019) Affine template matching by differential evolution with adaptive two-part search. *IEEJ Transactions on Electrical and Electronic Engineering* 14(4):615–622. <https://doi.org/10.1002/tee.22844>
- Sedelnikov A, Kurkin E, Quijada-Pioquinto JG, et al (2024) Algorithm for Propeller Optimization Based on Differential Evolution. *Computation* 12(3). <https://doi.org/10.3390/computation12030052>

- Shen Y, Liang J, Kang H, et al (2025) NLAPSMjSO-EDA: A Nonlinear Shrinking Population Strategy Algorithm for Elite Group Exploration with Symmetry Applications. *Symmetry* 17(2):153. <https://doi.org/10.3390/sym17020153>
- Song X, Yuan Z, Zhao M (2026) An adaptive differential evolution algorithm with a multi-probability selection mechanism guiding individual selection. In: 2026 International Conference on Generative Artificial Intelligence and Information Security (GAIS), pp 849–853, <https://doi.org/10.1109/GAIS69281.2026.11519130>
- Song Y, Wu D, Mohamed AW, et al (2021) Enhanced success history adaptive de for parameter optimization of photovoltaic models. *Complexity* 2021:6660115. <https://doi.org/10.1155/2021/6660115>
- Song Z, Tang Y, Ji J, et al (2020) Evaluating a dendritic neuron model for wind speed forecasting. *Knowledge-Based Systems* 201-202:106052. <https://doi.org/10.1016/j.knosys.2020.106052>
- Song Z, Sun L, Matsveichuk N, et al (2025) Diversity enhancement-based Differential Evolution with a novel perturbation strategy. *Swarm and Evolutionary Computation* 92:101822. <https://doi.org/10.1016/j.swevo.2024.101822>
- Song-Chol K, Sung-Nam K (2026) Ensemble of Two New Mutation Strategies for History-Based Parameter Adaptation Differential Evolution. *International Journal of Computational Intelligence and Applications* 25(1):2550011. <https://doi.org/10.1142/S1469026825500117>
- Stanovov V, Semenkin E (2023) Dual-Population Differential Evolution L-NTADE for ICSI-OC'2023 Competition. *Lecture Notes in Computer Science (including sub-series Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 13969 LNCS:516–525. https://doi.org/10.1007/978-3-031-36625-3_41
- Stanovov V, Semenkin E (2024a) Adaptation of the Scaling Factor Based on the Success Rate in Differential Evolution. *Mathematics* 12(4). <https://doi.org/10.3390/math12040516>
- Stanovov V, Semenkin E (2024b) Differential Evolution with Success Rate-based adaptation CL-SRDE for Constrained Optimization. In: 2024 IEEE Congress on Evolutionary Computation, CEC 2024 - Proceedings, pp 1–8, <https://doi.org/10.1109/CEC60901.2024.10612145>
- Stanovov V, Semenkin E (2024c) Success rate-based adaptive differential evolution L-SRTDE for CEC 2024 competition. In: 2024 IEEE Congress on Evolutionary Computation (CEC), pp 1–8, <https://doi.org/10.1109/CEC60901.2024.10611907>
- Stanovov V, Akhmedova S, Semenkin E (2020) Ranked Archive Differential Evolution with Selective Pressure for CEC 2020 Numerical Optimization. In: 2020 IEEE Congress on Evolutionary Computation (CEC), pp 1–7, <https://doi.org/10.1109/>

- Stanovov V, Akhmedova S, Semenkin E (2021) NL-SHADE-RSP algorithm with adaptive archive and selective pressure for CEC 2021 numerical optimization. In: 2021 IEEE Congress on Evolutionary Computation (CEC), IEEE, pp 809–816, <https://doi.org/10.1109/CEC45853.2021.9504959>
- Stanovov V, Akhmedova S, Semenkin E (2022) Dual-Population Adaptive Differential Evolution Algorithm L-NTADE. *Mathematics* 10(24). <https://doi.org/10.3390/math10244666>
- Storn R, Price K (1997) Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization* 11:341–359. <https://doi.org/10.1023/A:1008202821328>
- Sun G, Peng J, Zhao R (2018) Differential evolution with individual-dependent and dynamic parameter adjustment. *Soft Computing* 22(17):5747–5773. <https://doi.org/10.1007/s00500-017-2626-3>
- Sun G, Xu G, Gao R, et al (2023a) A fluctuant population strategy for differential evolution. *Evolutionary Intelligence* 16(6):1747–1765. <https://doi.org/10.1007/s12065-019-00287-6>
- Sun X, Jiang L, Shen Y, et al (2020) Success history-based adaptive differential evolution using turning-based mutation. *Mathematics* 8(9):1565. <https://doi.org/10.3390/math8091565>
- Sun Y, Cao Y (2026) An Improved Hybrid Evolutionary Algorithm with Sine-Based Parameter Adaptation. In: *Proceedings of 2026 International Conference on Blockchain Technology and Foundation Models, BTFM 2026*, pp 146–149, <https://doi.org/10.1109/BTFM69084.2026.11484512>
- Sun Y, Zhang K, Li Z, et al (2023b) A cascaded differential evolution optimization framework with adaptive population allocation and reduction. *Swarm and Evolutionary Computation* 82:101376. <https://doi.org/10.1016/j.swevo.2023.101376>
- Sun Y, Wu Y, Liu Z (2024) An improved differential evolution with adaptive population allocation and mutation selection. *Expert Systems with Applications* 258:125130. <https://doi.org/10.1016/j.eswa.2024.125130>
- Sun Y, Wu Y, Liu Z (2025) Improvement of differential evolution variants with nonlinear population adjustment and parameter adaption. *Expert Systems with Applications* 274:126945. <https://doi.org/10.1016/j.eswa.2025.126945>
- Tan D, Ding Z, Li J, et al (2023) Target-free vision-based approach for vibration measurement and damage identification of truss bridges. *Smart Structures and Systems* 31(4):421–436. <https://doi.org/10.12989/sss.2023.31.4.421>

- Tan Z, Li K, Wang Y (2021) Differential evolution with adaptive mutation strategy based on fitness landscape analysis. *Information Sciences* 549:142–163. <https://doi.org/10.1016/j.ins.2020.11.023>
- Tanabe R, Fukunaga A (2013) Success-history based parameter adaptation for differential evolution. In: 2013 IEEE congress on evolutionary computation, IEEE, pp 71–78, <https://doi.org/10.1109/CEC.2013.6557555>
- Tanabe R, Fukunaga A (2019) Reviewing and benchmarking parameter control methods in differential evolution. *IEEE transactions on cybernetics* 50(3):1170–1184. <https://doi.org/10.1109/TCYB.2019.2892735>
- Tanabe R, Fukunaga AS (2014) Improving the search performance of SHADE using linear population size reduction. In: 2014 IEEE congress on evolutionary computation (CEC), pp 1658–1665, <https://doi.org/10.1109/CEC.2014.6900380>
- Tang H, Lee J (2022) Adaptive initialization LSHADE algorithm enhanced with gradient-based repair for real-world constrained optimization. *Knowledge-Based Systems* 246:108696. <https://doi.org/10.1016/j.knosys.2022.108696>
- Tang Z, Peng L, Dai G, et al (2023) Enhancing the search ability of a hybrid LSHADE for global optimization of interplanetary trajectory design. *Engineering Optimization* 55(4):632–649. <https://doi.org/10.1080/0305215X.2021.2019250>
- Teo J (2006) Exploring dynamic self-adaptive populations in differential evolution. *Soft computing* 10(8):673–686. <https://doi.org/10.1007/s00500-005-0537-1>
- Tian M, Gao X (2019) Differential evolution with neighborhood-based adaptive evolution mechanism for numerical optimization. *Information Sciences* 478:422–448. <https://doi.org/10.1016/j.ins.2018.11.021>
- Tian M, Meng Y, He X, et al (2023) An Enhanced Adaptive Differential Evolution Algorithm With Multi-Mutation Schemes and Weighted Control Parameter Setting. *IEEE Access* 11:98854–98874. <https://doi.org/10.1109/ACCESS.2023.3312010>
- Tirronen V, Neri F (2009) Differential evolution with fitness diversity self-adaptation. In: Chiong R (ed) *Nature-inspired algorithms for optimisation*. Springer, Berlin, Heidelberg", p 199–234, https://doi.org/10.1007/978-3-642-00267-0_7
- Truong VH (2026) Optimizing nonlinear steel trusses using enhanced differential evolution with linear population reduction. *Steel and Composite Structures* 58(5):567–594. <https://doi.org/10.12989/scs.2026.58.5.567>
- Verma P, Chaudhary R, Lamba AK, et al (2025) ALSHADE-jSO Algorithm Solving the GNBG-II Test Suite : A Competition Entry on Numerical Global Optimization Competition on GNBG-II generated Test Suite at GECCO 2025. In: *GECCO 2025 Companion - Proceedings of the 2025 Genetic and Evolutionary Computation*

- Conference Companion, pp 9–10, <https://doi.org/10.1145/3712255.3735094>
- Viktorin A, Senkerik R, Pluhacek M, et al (2017) Distance based parameter adaptation for differential evolution. In: 2017 IEEE Symposium Series on Computational Intelligence, SSCI 2017 - Proceedings, pp 1–7, <https://doi.org/10.1109/SSCI.2017.8280959>
- Viktorin A, Senkerik R, Pluhacek M, et al (2018a) Cluster Occurrence in the DbL-SHADE Population. In: 2018 IEEE Congress on Evolutionary Computation, CEC 2018 - Proceedings, pp 1–8, <https://doi.org/10.1109/CEC.2018.8477793>
- Viktorin A, Senkerik R, Pluhacek M, et al (2018b) L-SHADE Algorithm with Distance Based Parameter Adaptation. Lecture Notes in Electrical Engineering 465:69–80. https://doi.org/10.1007/978-3-319-69814-4_7
- Viktorin A, Senkerik R, Pluhacek M, et al (2019a) Enhanced archive for SHADE. Advances in Intelligent Systems and Computing 837:40–55. https://doi.org/10.1007/978-3-319-97888-8_4
- Viktorin A, Senkerik R, Pluhacek M, et al (2019b) Distance based parameter adaptation for Success-History based Differential Evolution. Swarm and Evolutionary Computation 50:100462. <https://doi.org/10.1016/j.swevo.2018.10.013>
- Visutarrrom T, Chiang TC (2024) Economic dispatch using metaheuristics: Algorithms, problems, and solutions. Applied Soft Computing 150:110891. <https://doi.org/10.1016/j.asoc.2023.110891>
- Wang BC, Shui ZY, Feng Y, et al (2022) Evolutionary algorithm with dynamic population size for constrained multiobjective optimization. Swarm and Evolutionary Computation 73:101104. <https://doi.org/10.1016/j.swevo.2022.101104>
- Wang K, Gao S, Zhou M, et al (2025a) Fractional order differential evolution. IEEE Transactions on Evolutionary Computation 29(3):822–835. <https://doi.org/10.1109/TEVC.2024.3382047>
- Wang L, Huang Y, Ren J, et al (2024) Double-transfer learning-based least squares wavelet support vector machine for water quality classification. Desalination and Water Treatment 320:100714. <https://doi.org/10.1016/j.dwt.2024.100714>
- Wang SL, Foo Ng T, Keong BP, et al (2025b) Exploring Adaptive Population Sizing's Parameters in a Self-adaptive Differential Evolution. In: 2025 IEEE International Conference on Computing (ICOCO), pp 1–6, <https://doi.org/10.1109/ICOCO67189.2025.11334088>
- Wang T (2026) Co-Evolutionary Pruning and Dynamic Weight Aggregation Enhanced L-SHADE for Cuffless Blood Pressure Estimation. In: 2026 9th International Conference on Advanced Algorithms and Control Engineering (ICAACE), pp

- 1945–1948, <https://doi.org/10.1109/ICAACE69793.2026.11509202>
- Wang Y, Yi J, Liang Q, et al (2025c) Research on the Bidding Strategy of a Comprehensive Power Generation Group Encompassing Diverse Resources Based on Nested Game. In: IEEE Power and Energy Society General Meeting, pp 1–5, <https://doi.org/10.1109/PESGM52009.2025.11225800>
- Wei Z, Ding D, Zhou H, et al (2020) A flight maneuver recognition method based on multi-strategy affine canonical time warping. *Applied Soft Computing Journal* 95:106527. <https://doi.org/10.1016/j.asoc.2020.106527>
- Wen H, Zhou G, Wu K, et al (2025) Optimization Research on Magnetic Interference Parameter Identification and Compensation for AUV Platforms. *IEEE Access* 13:88509–88519. <https://doi.org/10.1109/ACCESS.2025.3568122>
- Wen H, Zhou G, Wu K, et al (2026) Research on AUV Magnetic Interference Parameter Identification Based on Geomagnetic Simulation Method and L-SHADE Algorithm. *IEEE Transactions on Instrumentation and Measurement* 75:1–18. <https://doi.org/10.1109/TIM.2026.3659660>
- Wong I, Liu W, Ho CM, et al (2017) Continuous Adaptive Population Reduction (CAPR) for Differential Evolution Optimization. *SLAS Technology* 22(3):289–305. <https://doi.org/10.1177/2472630317690318>
- Wu J, Yang H, Yang H, et al (2025) Multiframe Target Localization With Strap-down Seeker Based on Self-Adaptive L-SHADE. *International Journal of Aerospace Engineering* 2025(1):1619772. <https://doi.org/10.1155/ijae/1619772>
- Xia Q, Wang X, Tang L (2019) Furnace operation optimization with hybrid model based on mechanism and data analytics. *Soft Computing* 23(19):9551–9571. <https://doi.org/10.1007/s00500-018-3519-9>
- Xia X, Gui L, Zhang Y, et al (2021) A fitness-based adaptive differential evolution algorithm. *Information Sciences* 549:116–141. <https://doi.org/10.1016/j.ins.2020.11.015>
- Xu H, Lin S, Zhang Z, et al (2025) Differential evolution based on two-stage mutation strategy and multi-stage parameter control. *Applied Soft Computing* 180:113387. <https://doi.org/10.1016/j.asoc.2025.113387>
- Yadav S, Saha SK, Kar R, et al (2025) Evolutionary optimization-based descendent adaptive filter for noise confiscation in electrocardiogram signals. *Physical and Engineering Sciences in Medicine* 48(4):1911–1933. <https://doi.org/10.1007/s13246-025-01631-0>
- Yang G, Liu Z, Zhu Z (2021) Multi-level threshold segmentation based on LSHADE. In: 2021 International Conference on Digital Society and Intelligent Systems, DSInS

- 2021, pp 204–211, <https://doi.org/10.1109/DSInS54396.2021.9670556>
- Yang H, Zhou Z, Li Y (2023) A Cooperative Search Algorithm Based on Improved Probability Map of Target Acoustic Information for Multiple UUVs. *Journal of Unmanned Undersea Systems* 31(5):715–724. <https://doi.org/10.11993/j.issn.2096-3920.2022-0036>
- Yang Q, Yuan S, Gao H, et al (2024a) Differential evolution with migration mechanism and information reutilization for global optimization. *Expert Systems with Applications* 238:122076. <https://doi.org/10.1016/j.eswa.2023.122076>
- Yang X, Hu M, Huang G, et al (2024b) Multi-Layer Objective Model and Progressive Optimization Mechanism for Multi-Satellite Imaging Mission Planning in Large-Scale Target Scenarios. *Applied Sciences (Switzerland)* 14(19). <https://doi.org/10.3390/app14198597>
- Yang X, Zeng G, Cao Z, et al (2024c) Novel parameter identification for complex solar photovoltaic models via dynamic L-SHADE with parameter decomposition. *Case Studies in Thermal Engineering* 61:104938. <https://doi.org/10.1016/j.csite.2024.104938>
- Yang X, Zeng G, Cao Z, et al (2024d) Parameters estimation of complex solar photovoltaic models using bi-parameter coordinated updating L-SHADE with parameter decomposition method. *Case Studies in Thermal Engineering* 61:104917. <https://doi.org/10.1016/j.csite.2024.104917>
- Yao J, Chen Z, Liu Z (2021) Improved ensemble of differential evolution variants. *PLoS ONE* 16(8 August):1–22. <https://doi.org/10.1371/journal.pone.0256206>
- Ye X, Li J, Wang P, et al (2025) A comprehensive survey of adaptive strategies in differential evolutionary algorithms. *Swarm and Evolutionary Computation* 98:102081. <https://doi.org/10.1016/j.swevo.2025.102081>
- Yeh JF, Chen TY, Chiang TC (2019) Modified L-SHADE for Single Objective Real-Parameter Optimization. In: 2019 IEEE Congress on Evolutionary Computation, CEC 2019 - Proceedings, pp 381–386, <https://doi.org/10.1109/CEC.2019.8789991>
- Yildirim c, Bozkurt MH, Kahraman HT, et al (2025) Dental X-Ray image enhancement using a novel evolutionary optimization algorithm. *Engineering Applications of Artificial Intelligence* 142:109879. <https://doi.org/10.1016/j.engappai.2024.109879>
- Yin L, Li H, Ye Y, et al (2025) Deep convolutional generative adversarial network accelerated optimization algorithm for parameter optimization of permanent magnet synchronous generator controllers. *Applied Soft Computing* 180:113458. <https://doi.org/10.1016/j.asoc.2025.113458>

- Yu Y, Wang K, Zhang T, et al (2022) A population diversity-controlled differential evolution for parameter estimation of solar photovoltaic models. *Sustainable Energy Technologies and Assessments* 51. <https://doi.org/10.1016/j.seta.2021.101938>
- Zamuda A (2017) Adaptive constraint handling and Success History Differential Evolution for CEC 2017 Constrained Real-Parameter Optimization. In: 2017 IEEE Congress on Evolutionary Computation, CEC 2017 - Proceedings, pp 2443–2450, <https://doi.org/10.1109/CEC.2017.7969601>
- Zamuda A (2019a) Binary 100-digit challenge using IEEE-754 coded numerical optimization scenarios (100b-digit) and V-shape binary distance-based success history differential evolution (DISHv). In: GECCO 2019 Companion - Proceedings of the 2019 Genetic and Evolutionary Computation Conference Companion, pp 1821–1828, <https://doi.org/10.1145/3319619.3326898>
- Zamuda A (2019b) Function evaluations upto $1e+12$ and large population sizes assessed in distance-based success history differential evolution for 100-digit challenge and numerical optimization scenarios (DISHchain $1e+12$): A competition entry for "100-digit challenge, and four other numerical optimization competitions" at the genetic and evolutionary computation conference (GECCO) 2019. In: GECCO 2019 Companion - Proceedings of the 2019 Genetic and Evolutionary Computation Conference Companion, pp 11–12, <https://doi.org/10.1145/3319619.3326751>
- Zamuda A, Sosa JDH (2019a) Hot off the press in expert systems on underwater robotic missions: Success history applied to differential evolution for underwater glider path planning. In: GECCO 2019 Companion - Proceedings of the 2019 Genetic and Evolutionary Computation Conference Companion, pp 39–40, <https://doi.org/10.1145/3319619.3326763>
- Zamuda A, Sosa JDH (2019b) Success history applied to expert system for underwater glider path planning using differential evolution. *Expert Systems with Applications* 119:155–170. <https://doi.org/10.1016/j.eswa.2018.10.048>
- Zeng Z, Zhang M, Zhang H, et al (2022) Improved differential evolution algorithm based on the sawtooth-linear population size adaptive method. *Information Sciences* 608:1045–1071. <https://doi.org/10.1016/j.ins.2022.07.003>
- Zeng Z, Liang W, Gao L (2025) A differential evolution algorithm for numerical optimization based on the cosine-exponential population size adaptive method. *Memetic Computing* 17(2):19. <https://doi.org/10.1007/s12293-025-00445-7>
- Zhabitsky M, Zhabitskaya E (2013) Asynchronous differential evolution with adaptive correlation matrix. In: Proceedings of the 15th annual conference on Genetic and evolutionary computation, pp 455–462, <https://doi.org/10.1145/2463372.2463428>
- Zhan ZH, Wang ZJ, Jin H, et al (2020) Adaptive Distributed Differential Evolution. *IEEE Transactions on Cybernetics* 50(11):4633–4647. <https://doi.org/10.1109/TCYB.2020.3000000>

- Zhang G, Hou J, Wan C, et al (2022) Structural system identification and damage detection using adaptive hybrid Jaya and differential evolution algorithm with mutation pool strategy. *Structures* 46:1313–1326. <https://doi.org/10.1016/j.istruc.2022.10.130>
- Zhang J, Meng Z (2023) QUATRE-PM: QUasi-Affine TRansformation Evolution With Perturbation Mechanism. *IEEE Access* 11:88711–88729. <https://doi.org/10.1109/ACCESS.2023.3305925>
- Zhang X, Zhan ZH, Zhang J (2020) Adaptive Population Differential Evolution with Dual Control Strategy for Large-Scale Global optimization Problems. In: 2020 IEEE Congress on Evolutionary Computation, CEC 2020 - Conference Proceedings, pp 1–7, <https://doi.org/10.1109/CEC48606.2020.9185854>
- Zhang X, Liu Q, Qu Y (2023a) An adaptive differential evolution algorithm with population size reduction strategy for unconstrained optimization problem. *Applied Soft Computing* 138:110209. <https://doi.org/10.1016/j.asoc.2023.110209>
- Zhang XM, Yang JG, Feng QH, et al (2026) Collaborative optimization of well operations and adjustment strategies in waterflooding reservoirs using an enhanced adaptive differential evolution algorithm. *Petroleum Science* 23(5):2735–2757. <https://doi.org/10.1016/j.petsci.2026.03.030>
- Zhang Y, Peng L, Dai G, et al (2019) A population entropy based adaptation strategy for differential evolution. In: GECCO 2019 Companion - Proceedings of the 2019 Genetic and Evolutionary Computation Conference Companion, pp 330–331, <https://doi.org/10.1145/3319619.3321981>
- Zhang Y, Dai G, Peng L, et al (2023b) Enhancing differential evolution algorithm through a population size adaptation strategy. *Natural Computing* 22(2):379–392. <https://doi.org/10.1007/s11047-021-09855-1>
- Zhang Z, Zhu J, Nie F (2024) A novel hybrid adaptive differential evolution for global optimization. *Scientific Reports* 14(1):19697. <https://doi.org/10.1038/s41598-024-70731-w>
- Zhao F, Zhou H, Xu T, et al (2023) A self-learning differential evolution algorithm with population range indicator. *Expert Systems with Applications* 241:122674. <https://doi.org/10.1016/j.eswa.2023.122674>
- Zheng L, Luo S (2022) Adaptive Differential Evolution Algorithm Based on Fitness Landscape Characteristic. *Mathematics* 10(9). <https://doi.org/10.3390/math10091511>

- Zhong H, Li X, Gao L, et al (2021) Development of Admittance Control Method with Parameter Self-optimization for Hydraulic Series Elastic Actuator. *International Journal of Control, Automation and Systems* 19(7):2357–2372. <https://doi.org/10.1007/s12555-020-0465-y>
- Zhou B, Huang Y (2024) An adaptive archive differential evolution with non-linear population size reduction and selective pressure. *Information Sciences* 682:121273. <https://doi.org/10.1016/j.ins.2024.121273>
- Zhou G, Li L, Wu K, et al (2024) Numerical Simulation of Magnetic Interference Parameter Identification of AUV Based on L-SHADE Algorithm. *Binggong Xuebao/Acta Armamentarii* 45(8):2678–2687. <https://doi.org/10.12382/bgxb.2023.0599>
- Zhou W, Meng Z (2025) An adaptive differential evolution with dynamic perturbation and dimensional bidirectional crossover mechanism for diversity enhancement. *Engineering Applications of Artificial Intelligence* 141:109750. <https://doi.org/10.1016/j.engappai.2024.109750>
- Zhou W, Wang P, Zhao X, et al (2023) Anti-sine-cosine atom search optimization (ASCASO): a novel approach for parameter estimation of PV models. *Environmental Science and Pollution Research* 30(44):99620–99651. <https://doi.org/10.1007/s11356-023-28777-2>
- Zuo Y, Zhao F, Li Z (2021) A knowledge-based differential covariance matrix adaptation cooperative algorithm. *Expert Systems with Applications* 184:115495. <https://doi.org/10.1016/j.eswa.2021.115495>