symmetry

Article

Dual-Performance Multi-Subpopulation Adaptive Restart Differential Evolutionary Algorithm

Yong Shen , Yunlu Xie and Qingyi Chen *

School of Software, Yunnan University, Kunming 650000, China; sheny@ynu.edu.cn (Y.S.); xieyunlu@stu.ynu.edu.cn (Y.X.)
* Correspondence: devas9@ynu.edu.cn

Abstract: To cope with common local optimum traps and balance exploration and development in complex multi-peak optimisation problems, this paper puts forth a Dual-Performance Multi-subpopulation Adaptive Restart Differential Evolutionary Algorithm (DPR-MGDE) as a potential solution. The algorithm employs a novel approach by utilising the fitness and historical update frequency as dual-performance metrics to categorise the population into three distinct sub-populations: PM (the promising individual set), MM (the medium individual set) and UM (the un-promising individual set). The multi-subpopulation division mechanism enables the algorithm to achieve a balance between global exploration, local exploitation and diversity maintenance, thereby enhancing its overall optimisation capability. Furthermore, the DPR-MGDE incorporates an adaptive cross-variation strategy, which enables the dynamic adjustment of the variation factor and crossover probability in accordance with the performance of the individuals. This enhances the flexibility of the algorithm, allowing for the prioritisation of local exploitation among the more excellent individuals and the exploration of new search space among the less excellent individuals. Furthermore, the algorithm employs a collision-based Gaussian wandering restart strategy, wherein the collision frequency serves as the criterion for triggering a restart. Upon detecting population stagnation, the updated population is subjected to optimal solution-guided Gaussian wandering, effectively preventing the descent into local optima. Through experiments on the CEC2017 benchmark functions, we verified that DPR-MGDE has higher solution accuracy compared to newer differential evolution algorithms, and proved its significant advantages in complex optimisation tasks with the Wilcoxon test. In addition to this, we also conducted experiments on real engineering problems to demonstrate the effectiveness and superiority of DPR-MGDE in dealing with real engineering problems.

Keywords: differential evolution; dual performance; sub-population; restart strategy

check for updates

Academic Editors: Zhixun Su and Theodore E. Simos

Received: 15 December 2024
Revised: 20 January 2025
Accepted: 31 January 2025
Published: 3 February 2025

Citation: Shen, Y.; Xie, Y.; Chen, Q. Dual-Performance Multi-Subpopulation Adaptive Restart Differential Evolutionary Algorithm. *Symmetry* **2025**, *17*, 223. https://doi.org/10.3390/sym17020223

1. Introduction

Since its proposal, the DE algorithm has shown significant advantages in terms of global search capability, parameter setting, convergence speed, robustness, implementation difficulty and applicability, which makes the DE algorithm stand out among many meta-heuristic optimisation algorithms and has been widely employed in high-dimensional and complex optimisation problems [1]. In a multitude of application scenarios, including engineering design [2,3], hyper-parameter optimisation in machine learning [4,5] and pattern recognition in data analysis [6,7], DE algorithms have demonstrated effective solution results due to their robust search capability and adaptability. However, conventional DE algorithms frequently encounter obstacles [8] during the optimisation procedure, particularly when confronted with multi-peaks and high-dimensional complex functions. They are prone to being constrained by local optima and slow to converge, which ultimately impacts the final optimisation outcome.

One of the primary challenges associated with high-dimensional multi-peak problems is the need to achieve an optimal balance between global exploration [9] and local exploitation. It is common for traditional DE algorithms to be susceptible to what is known as the "local optimality trap" in such circumstances. This refers to a situation in which the algorithm converges in a local region but fails to identify the global optimal solution. Furthermore, the algorithm's global search capability is further diminished due to the gradual decline in population diversity over the course of iterations, which renders it challenging to surpass the current solution limit. Furthermore, the convergence speed of the differential evolutionary algorithm frequently proves inadequate when confronted with high-dimensional complex problems, particularly in the latter stages of convergence. This is due to the absence of a meticulous adaptive mechanism, which makes it challenging to strike a balance between the precision and velocity of the solution. The existence of these issues has led to the prioritisation of enhancing the global search capability of differential evolutionary algorithms [10] and accelerating their convergence speed [11] as key objectives in optimisation algorithm research in recent years.

In order to address these challenges, numerous researchers have put forth a multitude of proposed enhancements to differential evolutionary algorithms. The first strategy is adaptive parameter tuning [12], which is a common approach to enhancing the flexibility of the search process by dynamically adjusting the parameters of the algorithm at different optimisation stages. To illustrate, the adaptive differential evolutionary algorithm (JADE) [13] enhances the convergence velocity to a certain degree by means of adaptively modifying the variance factor and crossover probability. The literature [14] proposes a new method, Joint Adaptation of Parameters in DE (JADE), which is based on the core idea of dynamically updating the selection probabilities of parameter generating function pairs in accordance with the feedback obtained from the search process. The algorithm introduces the Rank-based Adaptation (RAM) method, which learns multiple probability distributions through the interval of fitness ranking. This is coupled with JADE to form RAM-JADE, which simultaneously adapts the selection probabilities of the control parameter pairs and the variation strategies. APDDE [15] generates the population of progeny by introducing the detected values in the iteration and integrating them into the two variation strategies, thus retaining the performance of the optimal parameter values. Fan [16] proposed an Adaptive DE Algorithm with Discrete Mutation Control Parameters (DMPSADE). In DMPSADE, each variable of each individual is associated with a distinct variation control parameter, and each individual is linked to a unique crossover control parameter and variation strategy. Nevertheless, although adaptive parameter adjustment enhances flexibility, it is typically accompanied by an increased computational overhead, and its performance is more sensitive to the initial parameters. Furthermore, the effect is not stable across different tasks.

The multiple subpopulation structure [17] provides another way of thinking about the search–exploitation balance. This strategy increases the effectiveness of the search by dividing the population into subpopulations, each of which performs different optimisation tasks. For example, Wu [18] studied and proposed a Multi-population Integrated Differential Evolutionary Algorithm (MPEDE) that integrates three mutation strategies. The algorithm divides the population into three small indicator subpopulations and a larger reward subpopulation, each using a different mutation strategy. The performance of each mutation strategy is periodically evaluated and the reward subpopulation is dynamically assigned to the best performing strategy, while the control parameters of each mutation strategy are independently adjusted. The study [19] proposes a Self-adaptive Multi-population Differential Evolution Algorithm called SAMDE, wherein the population is randomly divided into three subpopulations of equal size, each with a different mutation strategy. At the end of each generation, all subpopulations are independently updated and recombined. The algorithm [20] proposes a co-evolutionary mutation strategy for the elite population with three mutation strategies, adopts a reverse learning mechanism to generate the initial few subpopulations, and uses a new multi-population parallel control strategy to maintain the search efficiency of the subpopulations. MMDE [21] is an Adaptive Multi-population-based Differential Evolutionary Population Algorithm that designs a population division strategy using fitness and Euclidean distance as the basis for judging individual potential, introduces an appropriate mutation strategy along the parameter control strategy and assigns it to each subpopulation for evolution to achieve balanced evolution. Sun [22] proposed a Two-stage Differential Evolutionary Algorithm with Mutation Strategy Combinations (TS-MSCDE). The algorithm divides the whole population into two symmetric sub-populations in the order of fitness from smallest to largest, and assigns four mutation strategies with different search behaviours to the superior and inferior sub-populations in two phases according to the mechanism of heterogeneous two-stage bipartite sub-population. However, the introduction of a multiple subpopulation structure also introduces some complexity, especially in high-dimensional problems, where cooperation and proper partitioning between subpopulations is difficult to control, leading to possible instability in the algorithm's performance.

In addition, restart mechanisms are widely used to avoid local optimisation. By reinitialising the population distribution when the algorithm becomes stagnant, the restart strategy can improve the ability of the algorithm to jump out of the local optimum, e.g., ADE-DMRM [23] (Adaptative Differential Evolution with Enhanced Diversity and Restart Mechanism) proposes a novel restart mechanism that identifies currently stagnant individuals by combining a stagnation tracker and a diversity evaluation metric, and then regenerates them using a dimensionality learning-based approach. The study [24] proposed an adaptive differential evolutionary algorithm (ADEDMR) based on a depth-informed mutation strategy and a restart mechanism, which uses a novel population restart mechanism to further enhance the population diversity by adaptively improving the search ability of the hopeless individuals and randomly replacing some poorer individuals with wavelet walks. Tian [25] et al. proposed a new restart mechanism that identifies the current stagnant individuals by using the superior individuals to search the hyper-rectangle and replacing the inferior individuals with randomly generated probabilities from the search space to improve the population diversity and use the useful information of the superior individuals. DPMADE [26] proposed a new restart strategy to improve the search performance of the algorithm by replacing the individuals with randomly generated individuals by Gaussian walks that are meaningless in terms of both fitness value and historical updates.

In summary, existing methods have made many advances in optimising differential evolution algorithms, but they still have deficiencies in adaptive balancing, maintaining population diversity and global jumping out of local optimums. Therefore, this paper proposes a dual-performance multi-subpopulation adaptive restart differential evolution algorithm (DPR-MGDE) to overcome the shortcomings of traditional DE algorithms. The main works and innovations are clearly divided into the following points:

1. Proposing a population division mechanism based on dual-performance indicators: Based on the adaptability and renewal frequency of individuals, a dual-performance indicator division method is proposed to divide the population into three subgroups: PM, MM, and UM, and to make different groups perform different duties according

to their development potential, so as to achieve the purpose of balanced exploration and development.

2. Use of adaptive cross-variation mechanism: The variation factor and crossover probability are dynamically adjusted according to the performance of individuals to achieve a more flexible search strategy. Individuals with excellent performance are given smaller variance factors and larger crossover probabilities to focus on local exploitation, while individuals with poor performance are given larger variances to enhance global exploration.
3. Introduction of a collision-based Gaussian wandering restart strategy: a collision-based Gaussian wandering restart mechanism is triggered when the actual collision rate exceeds a dynamically adjusted threshold. This mechanism restarts only the UM sub-population (un-promising individuals) to reduce the extra overhead of restarting the entire population. The spread of Gaussian wandering is dynamically adjusted, which helps to enhance the exploration capability at the beginning of the algorithm and the exploitation capability at a later stage, being able to improve the possibility of jumping out of the local optimum at a later stage by increasing the diversity of the population.
4. Validation of the algorithm: The performance of DPR-MGDE in the CEC2017 benchmark and the real engineering problems are experimentally tested, which shows that the algorithm outperforms newer variants of differential evolution algorithms in terms of both solution and accuracy. The effectiveness of DPR-MGDE in complex optimisation tasks is also verified, especially in terms of global exploration, optimisation accuracy and maintaining population diversity, where it shows significant advantages.

The following is the organisational structure of this paper:

1. Section 1, Introduction. It mainly describes the background, significance, motivation and content of this paper, and also outlines the research progress of differential evolutionary algorithms.
2. Section 2, Related Work. It mainly introduces the research on traditional DE algorithms, multi-subpopulation differential evolutionary algorithms and restart mechanisms.
3. Section 3, our proposed DPR-MGDE. mainly details the contents of DPR-MGDE, and gives the related pseudo-code and flowchart.
4. Section 4, Experiments. The data of parameter setting experiments of DPR-MGDE, comparison experiments with newer improved differential evolution algorithms and experiments on real engineering problems are mainly given.
5. Section 5, Summary. An overview summary of DPR-MGDE is given, and an outlook for future work on DPR-MGDE is given.

2. Related Works

In this section, we begin with an overview of traditional differential evolution (DE) algorithms, followed by a discussion of research related to subpopulation delineation, and an introduction to the restart mechanism immediately thereafter.

2.1. Classical DE Algorithm

The main goal of DE is to find the optimal solution that minimises (or maximises) the objective function under the given constraints by means of continuous evolution. DE is commonly used to solve global optimisation problems, especially when the search space is complex and contains multiple local optimal solutions. An optimisation problem in general form can be expressed as follows:

$$\arg\min_{\vec{X}\in\Omega} f(\vec{X}) \tag{1}$$

where $f(\vec{X})$ is the objective function, $\vec{X}$ is the solution vector and Ω is the search space. The core idea is based on the concept of "differential variation", i.e., the difference vector between two solutions is used to generate new solutions that provide direction for the search. The difference between two solutions is used to generate new solutions, giving direction to the search. The difference between solutions in the population is used to gradually approach the global optimal solution so that the optimisation process can be completed with low computational complexity. The execution of the DE algorithm consists of the following steps: initialisation, mutation, crossover and selection. Each step is described in detail below.

2.1.1. Population Initialisation

In differential evolutionary algorithms, the population must first be initialised and generated by a uniform distribution [27]. The population size is usually denoted by NP. Each individual is a dimension vector, where is the dimension of the problem. Each dimension of an individual has a specified upper and lower bound, where D is the dimension of the problem. The purpose of initialisation is to randomly distribute the individuals throughout the search space to ensure the diversity of the population.

$$x_{ij} = L_j + rand(1,0) \cdot (U_j - L_j) \tag{2}$$

where x_{ij} is the value of the jth dimension of the ith individual and denotes a uniform random number in the range. In this way, each individual in the population is initially randomly generated in the search space.

2.1.2. Mutation Operations

The variation operation is a central step in differential evolution and is used to generate a variation vector for each individual, helping the population to maintain diversity and avoid falling into local optima. Mutation generates new individuals by combining the difference vectors between solutions in the population and other solutions. There are various common variation strategies, and the following are a few classical ones:

DE/rand/1
$$\vec{v}_i = \vec{x}_{r1} + F \cdot (\vec{x}_{r2} - \vec{x}_{r3}) \tag{3}$$

DE/best/1
$$\vec{v}_i = \vec{x}_{\text{best}} + F \cdot (\vec{x}_{r1} - \vec{x}_{r2}) \tag{4}$$

DE/current-to-best/1
$$\vec{v}_i = \vec{x}_i + F \cdot (\vec{x}_{\text{best}} - \vec{x}_i) + F \cdot (\vec{x}_{r1} - \vec{x}_{r2}) \tag{5}$$

DE/rand/2
$$\vec{v}_i = \vec{x}_{r1} + F \cdot (\vec{x}_{r2} - \vec{x}_{r3}) + F \cdot (\vec{x}_{r4} - \vec{x}_{r5}) \tag{6}$$

DE/best/2
$$\vec{v}_i = \vec{x}_{\text{best}} + F \cdot (\vec{x}_{r1} - \vec{x}_{r2}) + F \cdot (\vec{x}_{r3} - \vec{x}_{r4}) \tag{7}$$

2.1.3. Crossover Operation

The crossover operation combines the variation vector generated by the mutation with the current individual to generate a "test vector". This process controls the proportion of new individuals that inherit the variant vector and the components of the current individual to increase population diversity.

Binary crossover is the most commonly used crossover method in DE, by randomly selecting each component of an individual and deciding whether it is to be inherited from the current individual or from the variation vector. The rule for generating the test vector u_{ij} is as follows:

$$u_{ij} = \begin{cases} v_{ij} & \text{if } \mathrm{rand}(0,1) \leq CR \text{ or } j = j_{\mathrm{rand}} \\ x_{ij} & \text{otherwise} \end{cases} \tag{8}$$

where CR is the crossover probability, which controls the probability that a component in the variant vector enters the test vector, and j_{rand} is a dimension chosen randomly to ensure that at least one component comes from the variant vector.

2.1.4. Select Operation

The purpose of the selection operation is to ensure that the population gradually evolves to a more optimal solution, i.e., individuals with higher fitness are retained into the next generation. The selection mechanism compares the trial vector $\vec{u}_i$ with the fitness value of the current individual $\vec{x}_i$:

$$\vec{x}_i^{(g+1)} = \begin{cases} \vec{u}_i & \text{if } f(\vec{u}_i) < f(\vec{x}_i) \\ \vec{x}_i & \text{otherwise} \end{cases} \tag{9}$$

For the minimisation problem, if the fitness of the test vector is better than that of the current individual, the current individual is replaced with the test vector; otherwise, the current individual is kept unchanged.

In traditional DE algorithms, the choice of variation strategy is a key factor affecting the performance of the algorithm. Different variation strategies help to find an appropriate balance between exploration and exploitation and to adapt to the needs of different optimisation problems. In addition, the crossover probability CRC and the variation factor F are also key parameters that together determine the convergence speed and global search capability of the DE algorithm.

The interaction of these steps makes the DE algorithm robust and adaptable, which is widely used in real-world complex optimisation problems. By adjusting the variation strategy and parameter settings, the DE algorithm can flexibly switch between global search and local exploitation, showing significant advantages in multi-peak and non-linear problems.

2.2. Differential Evolutionary Algorithms for Multinomial Populations

In order to balance the ability to explore and exploit, many scholars have proposed to divide the population into multiple sub-populations and to carry out their respective mutation strategies independently, and this approach is now widely used by everyone.

In paper [18], the population is divided into three equal-sized indicator sub-populations and one larger reward sub-population. Each indicator sub-population corresponds to one mutation strategy, namely "current-to-pbest/1", "current-to-rand/1" and "rand/1". During each generation of evolution, the current best-performing mutation strategy is identified based on the ratio of fitness improvement to depletion function evaluation, where depletion function evaluation is the function evaluations consumed by the jth mutation strategy during the former ng generations, and the rewarding subpopulation is dynamically assigned to this strategy. This design allows for better-performing mutation strategies to receive more computational resources, thus adaptively optimising the algorithm performance. Li et al. [28] proposed a new differential evolutionary algorithm that combines multiple sub-populations and an elite regeneration mechanism. Each sub-population may use different mutation strategies or parameter settings to improve the algorithm's search ability and adaptability. LMOMMDE [29] proposed a differential evolutionary algorithm based on multiple sub-populations and multiple strategies for large-scale multi-objective optimisation problems. The population is divided into three sub-populations of different ranks based on individual characteristics, and the advantage of multi-population strategy is utilised to maintain population diversity. Multiple mutation strategies are introduced into the subpopulations of different ranks to balance the diversity and convergence of the subpopulation individuals. In addition, the timing of re-population is determined according to the evolutionary state of the subpopulation to ensure that individuals are fully evolved within their own population, while information is exchanged under certain conditions. The differential evolutionary algorithm for multi-subpopulations shows significant advantages and potentials in solving various optimisation problems through its flexible strategy selection, effective diversity maintenance mechanism and intelligent resource allocation.

2.3. *Differential Evolutionary Algorithms with Restart Mechanisms*

In the process of the DE algorithm, it is easy to fall into local optimum. To solve this problem, the restart mechanism [27] is proposed as an effective solution strategy. The study [30] increases the population size and restarts when search stagnation is detected. A restart will be triggered when the difference between the maximum and minimum of the population individuals is less than, or the difference between the maximum and minimum fitness of an individual is small. Poláková [31] and others proposed a strategy to trigger a restart by iterating the stagnator and the Euclidean distances of the population individuals, which they attributed to the smaller diversity of the population and/or the convergence to the local minima in which the population is trapped.

In summary, the restart mechanism is designed to solve the local optimum problem, which is caused by the lack of population diversity. However, restarting too early may prevent reaching a good local optimum, while restarting too late means that the search effort is wasted when the population is already stuck.

At the Evolutionary Computation Conference in 2022, Kitamura et al. [32] argued that collision frequency can be successfully used as an indicator of search stagnation, and proposed that the diversity of a population is correlated with the collision rate, and that the more homogeneous a population is, the greater the likelihood that very similar individuals will be selected during reproduction, leading to collisions. This approach sets up the population as a hash table, stores the generated individuals as keys and fitness as a value and detects whether a collision has occurred by checking whether a newly generated individual is already in the hash table.

3. The Proposed DPR-MGDE

In this section, the proposed Dual-Performance Multi-Group Adaptive Restart Differential Evolution (DPR-MGDE) is presented in detail. The algorithm integrates dual-performance metrics, multiple subgroup partitioning, adaptive mutation strategy and collision-based Gaussian wandering restart mechanism to address the limitations of traditional DE algorithms. The following subsections describe each component of the proposed algorithm.

3.1. *Dual-Performance Indicator*

Most of the differential evolutionary algorithms are judged by the fitness, which directly reflects the optimisation goal of the objective function, facilitates the comparison of the strengths and weaknesses of individuals and is simpler to compute and implement.

However, the fitness only reflects the value of the objective function of an individual in the current generation, ignoring the dynamic performance of the individual, and may mislead the algorithm to focus on the local optimal region in multi-peak problems. Therefore, additional mechanisms (e.g., update frequency or dual-performance metrics) may be needed to assist. DPR-MGDE differs from the judging metrics used by the MMDE algorithm in the article [21] in that MMDE uses fitness with Euclidean distance as a dual-performance metric, with Euclidean distance placing more emphasis on the relative positions of the individuals, which makes it suitable for fast convergence scenarios. We use the fitness (function value) and the number of individual history updates (the number of times a mutant individual replaces the original individual) as dual-performance metrics, according to which the size of individual development potential is judged, which is more suitable for dynamically changing optimisation problems or multi-peak optimisation problems. By focusing on the frequency of updates, the population can be effectively prevented from falling into a local optimum, especially for the retention of dynamically active individuals with low fitness. The dual-performance index is calculated as follows:

$$P_{metric,i} = 0.5 \cdot \frac{mean(f)}{f_i + \epsilon} + 0.5 \cdot \frac{U_i}{mean(U) + \epsilon} \tag{10}$$

where f_i is the fitness value of the ith individual, which indicates its performance on the optimisation objective, U_i is the historical update frequency of the ith individual, which indicates the number of times an individual has been replaced or improved during the evolution process, $mean(f)$ is the mean of the fitness values in the current population, $mean(f)$ is the mean of the update frequency in the current population and the inclusion of minima is used to prevent the denominator from being zero.

3.2. Subpopulation Delineation Based on Dual-Performance Metrics

Based on the above dual-performance indicators, we propose a sub-population division strategy based on dual-performance indicators, aiming at a reasonable division of the population based on the static adaptive performance and dynamic update frequency of individuals. Firstly, we need to calculate $mean(f)$ and $mean(U)$; next, the population is divided into the potential subpopulation $PM = \{i \mid f_i < mean(f) \wedge U_i > mean(U)\}$, which contains individuals with good adaptation and dynamic activity, the medium subpopulation $MM = \{i \mid f_i < mean(f) \vee U_i > mean(U)\} \setminus PM$, which contains individuals with good adaptation or dynamic activity and the inferior subpopulation $UM = \{i \mid f_i > mean(f) \wedge U_i < mean(U)\}$, which contains individuals with poor adaptation and dynamic inactivity. Different sub-populations undertake different optimisation tasks, with the PM population focusing on exploring new solutions, the MM population on intensive exploitation and the UM population on maintaining diversity. According to this approach, the population division can flexibly adjust the composition of sub-populations with the changes in the optimisation stage, reasonably allocate resources during the optimisation process and realise the co-evolution of the populations, so as to improve the overall performance of the algorithm.

However, when the algorithm just enters the evolutionary stage, the update times of individuals U_i are all 0; then, the above sub-population division strategy cannot be implemented. Therefore, in order to avoid this problem, before the number of iterations of the algorithm is less than 10% of the maximum number of iterations, we use the sub-population division strategy in MMDE [21] for the population, which is based on the fitness and the Euclidean distance between the individual and the optimal individual, which avoids the above problem.

3.3. Adaptive Mutation Crossover Strategies

In DPR-MGDE, we will adopt three different mutation strategies for the three sub-populations.

The PM sub-population contains the individuals with the highest potential in the current population, and its main task is to explore new solutions and expand the search space; therefore, we improve the DE/rand/1 strategy with the following individual generation of mutations:

$$\vec{v}_i = \vec{x}_i + F_i \cdot \left(\vec{x}_{guider} - \vec{x}_i\right) \tag{11}$$

where $\vec{x}_{guider}$ is the most promising individual in the current population, the one with the highest score on the dual performance indicator $P_{metric,guider}$, called the guider, and the strategy emphasises on the offset of individuals towards the guider so as to explore further regions of the search space with the help of the guider's excellent performance.

The MM sub-population consists of individuals whose fitness or update frequency is higher than the mean, but their potential and performance are not as good as the PM population, and their main task is to develop the current optimal solution in depth and find a better solution, so the strategy used by the MM population is an improved strategy based on DE/current-to-best/1, and its individuals generate the amount of mutation in the following way:

$$\vec{v}_i = \vec{x}_i + F_i \cdot \left(\vec{x}_{guider} - \vec{x}_i\right) + F_i \cdot (\vec{x}_{r1} - \vec{x}_{r2}) \tag{12}$$

where $\vec{x}_{r1}$, $\vec{x}_{r2}$ is a randomly selected individual from NP, the strategy replaces the current population optimal individual $\vec{x}_{best}$ in DE/current-to-best/1 with the guider $\vec{x}_{guider}$, which increases the guidance of the two random individual difference vectors while moving closer to the guider, improving the exploitation efficiency and reducing the possibility of falling into the local optimum through the combination of moderate guidance and randomness.

UM sub-populations consist of individuals whose fitness and update frequency are below the mean. Their task is to maintain the diversity of the population and to avoid the population falling into a single optimisation direction, and their individuals generate mutations in the following way:

$$\vec{v}_i = \vec{x}_i + F_i \cdot (\vec{x}_{r1} - \vec{x}_{r2}) + F_i \cdot (\vec{x}_{r3} - \vec{x}_{r4}) \tag{13}$$

where $\vec{x}_{r1}$, $\vec{x}_{r2}$, $\vec{x}_{r3}$, $\vec{x}_{r4}$ is a randomly selected individual from the NP, and this strategy no longer uses $\vec{x}_{guider}$ and adds more random difference vectors to further increase population diversity.

We implement a different variation strategy for each sub-population so that it focuses on different aspects, which helps to balance the ability between DPR-MGDE development and exploration, ensuring that an optimal solution can be obtained in the search space while taking into account the diversity of the population. With the variance factor F_i and the crossover factor CR_i in the above variance strategy, we employ adaptive tuning of dynamic parameters based on P_{metric} as follows:

$$F_i = F_{\min} + (F_{\max} - F_{\min}) \cdot \left(1 - \frac{P_{\text{metric},i}}{\max(P_{\text{metric}})}\right) \tag{14}$$

$$CR_i = CR_{\min} + (CR_{\max} - CR_{\min}) \cdot \frac{P_{\text{metric},i}}{\max(P_{\text{metric}})} \tag{15}$$

where F_{min} and F_{max} are the minimum and maximum values of the variance factor. When $P_{metric,i}$ tends to the maximum value of the current population, F_i is close to F_{min}, which emphasises local exploitation, and when $P_{metric,i}$ is small, F_i is close to F_{max}, which enhances the global search capability. CR_{min} and CR_{max} are the maximum and minimum values of the crossover factor. When $P_{metric,i}$ tends to the maximum value of the current population, CR_i is close to CR_{max}, it emphasises variable exchange with high probability and is suitable for local search, and when $P_{metric,i}$ is small, CR_i is close to CR_{min}, which is suitable for global search.

In the early stage of optimisation, when the population needs to search extensively for the global optimum, the larger F_i and smaller CR_i help to enhance the exploration ability, and in the late stage of optimisation, when the population is close to convergence, the smaller F_i and larger CR_i help to accelerate the convergence speed. Through adaptive parameter tuning based on dual-performance metrics, DPR-MGDE achieves responsiveness to the demands of different optimisation stages. This dynamic adjustment strategy not only enhances the flexibility of the algorithm, but also significantly improves the exploration and exploitation balance of the population, ensuring the efficiency and robustness of the algorithm in complex optimisation problems.

3.4. Collision-Based Gaussian Wandering Restart Mechanism

In this paper, we propose a collision-based Gaussian wandering restart strategy. In this strategy, we use the collision frequency mentioned in Section 2.3 as the condition to trigger the restart: in the process of evolution, individuals will tend to evolve in the direction of the optimal individual, which will lead to the homogenisation of the population; at this time, the similarity of the individuals in the population becomes higher and higher and the diversity of the population becomes lower and lower, and when two individuals are the same, we define this phenomenon as collision. When the collision frequency between individuals becomes higher and higher, the evolution of the population will be problematic, and when the collision frequency reaches a threshold, a restart will be triggered, and the individuals in the UM will undergo a Gaussian wandering to produce new individuals to replace the original ones, thus increasing the diversity of the population and helping the algorithm to jump out of the local optimum.

We will use the collision frequency [32] as a judgement of whether to trigger the restart mechanism. In optimisation algorithms, maintaining population diversity is crucial to avoid premature convergence. Dynamically adjusting the collision rate threshold allows the algorithm to balance global search and local exploitation by keeping diversity high at the beginning while focusing on searching near the optimal solution at a later stage. Therefore, the collision rate threshold is calculated as follows:

$$C = C_{initial} + (C_{final} - C_{initial}) \cdot \frac{Iter}{MaxIter} \tag{16}$$

where C is the collision frequency threshold, which is jointly influenced by the initial collision frequency threshold and the final collision frequency threshold, and the value increases gradually with the number of iterations, transitioning from lower thresholds to maintain diversity to higher thresholds to facilitate local search, balancing exploration and exploitation.

When the individuals in the population are too concentrated and the collision frequency exceeds the threshold, a restart mechanism will be performed to restore the diversity of the population and avoid local optimality. We will perform a Gaussian wandering [33] restart for the UP sub-population (inferior individuals in the population) to generate new individuals guided by the global optimal solution. The Gaussian wandering formula is as follows:

$$y_i' = Gaussian(y_{best}, \sigma) + (rand \cdot y_{best} - rand \cdot y_i) \tag{17}$$

$$\sigma = \left| \frac{\log(Iter + 1)}{Iter + 1} \cdot (y_i - y_{best}) \right| \tag{18}$$

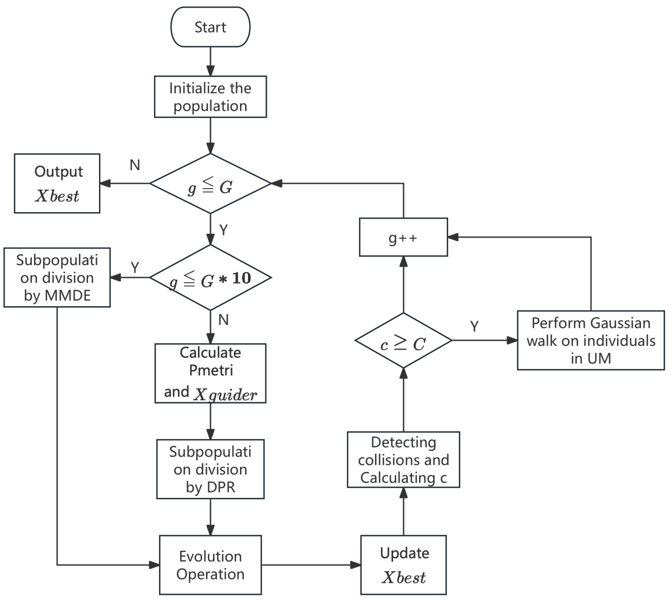
where y_i' is the generated new individual, y_{best} is the optimal solution of the current population and σ is the diffusion range of dynamic adjustment. As the number of iterations increases, σ will decrease, thus reducing the scope of search and focusing on local development, and if the difference between the individual and the optimal solution is large, the value of σ will increase accordingly to increase the scope of search. Different from other ways of generating new individuals, this method provides reasonable updates for individuals with unsatisfactory results, which can effectively eliminate useless individuals in the population, thus improving the search ability of the algorithm. The pseudo-code for this mechanism is as follows Algorithm 1:

Algorithm 1 Collision-based Gaussian Walk Restart Mechanism

```
Input: $population, fitness, hash\_table, UM, C\_initial, C\_final$
Output: updated population
 1: Calculate collision rate threshold C by Equation (16)
 2: Calculate current collision_rate
 3: while $g \leq G$ do
 4:   if $collision\_rate > C$ then
 5:     for $i = 1 : UM$ do
 6:       Perform Gaussian walk restart on UM subpopulation by Equation (17)
 7:       Assess the fitness of new individuals new_fitness
 8:       if $new\_fitness < fitness$ then
 9:         Update individual
10:       end if
11:     end for
12:   end if
13:   g++
14: end while
```

3.5. DPR-MGDE

Based on the above points, this paper proposes the dual-performance multi-subpopulation adaptive differential evolutionary algorithm, which divides the subpopulations by dual performance and improves the traditional mutation strategy by using a dual-performance bootstrap, in addition to introducing a collision-based Gaussian wandering restart strategy, whose pseudo-code and flowchart are as follows Algorithm 2 and Figure 1:



Start
Initialize the population
$g \leqq G$
N
Output $Xbest$
Y
$g \leqq G*10$
Y
Subpopulation division by MMDE
N
Calculate Pmetri and X_{guider}
Subpopulation division by DPR
Evolution Operation
Update $Xbest$
Detecting collisions and Calculating c
$c \geq C$
Y
Perform Gaussian walk on individuals in UM
g++

Figure 1. DPR-MGDE flowchart.

Algorithm 2 DPR-MGDE

```
Input: $NP, G_{max}, F_{min}, F_{max}, CR_{min}, CR_{max}, c, C, P$
1:  Set the current generation $g = 0$, initialise the population $P$ and evaluate their fitness
2:  while $g \leq G_{max}$ do
3:    Evaluate $P_{metri}$ by Equation (10) and select the $x_{guider}$
4:    if $g \leq G_{max} * 10\%$ then
5:      Subpopulation division by MMDE [21]
6:    else
7:      Subpopulation division by DPR
8:    end if
9:    for each individual $x_i$ in $PM$ do
10:     Evaluate $F_i$ and $CR_i$ by Equations (14) and (15)
11:     Generate a mutation vector $v_i$ of $x_i$ by Equation (11)
12:     Perform cross selection operation
13:     Check for collisions and update collision frequency
14:   end for
15:   for each individual $x_i$ in $MM$ do
16:     Evaluate $F_i$ and $CR_i$ by Equations (14) and (15) and randomly choose $x_1$, $x_2$
        from population
17:     Generate a mutation vector $v_i$ of $x_i$ by Equation (12)
18:     Perform cross selection operation
19:     Check for collisions and update collision frequency
20:   end for
21:   for each individual $x_i$ in $UM$ do
22:     Evaluate $F_i$ and $CR_i$ by Equations (14) and (15) and randomly choose $x_1$, $x_2$, $x_3$, $x_4$
        from population
23:     Generate a mutation vector $v_i$ of $x_i$ by Equation (13)
24:     Perform cross selection operation
25:     Check for collisions and update collision frequency
26:   end for
27:   Evaluate $c$ by Equation (16)
28:   while $c > C$ do
29:     for each individual $x_i$ in $UM$ do
30:       Gaussian walk generates new individuals by Equation (17)
31:     end for
32:   end while
33:   Update current optimal solution
34:   Set $g = g + 1$
35: end while
Output: The best individual and its fitness value.
```

In order to illustrate the feasibility of DPR-MGDE, we perform complexity analysis on it. The time complexity of the DPR-MGDE algorithm mainly consists of the following parts:

1. Population initialisation: The time complexity of initialising the population is $O(NP \times D)$, where NP is the population size and D is the problem dimension.
2. Subpopulation division: The time complexity of subpopulation division mainly depends on the population size NP and the problem dimension D. Specifically, the algorithm needs to calculate the fitness and historical update frequency of each individual, and then divide the population into three subpopulations (PM, MM, UM) based on these indicators. The time complexity of this process is $O(NP \times D)$. In each iteration, the subpopulation division operation needs to be repeated, so the time complexity of subpopulation division in the entire iteration process is $O(NP \times D \times G)$.

3. Iteration process: In each iteration, the algorithm needs to perform mutation, crossover and selection operations on each individual. The time complexity of these operations is $O(NP \times D)$. Since the number of iterations of the algorithm is G, the time complexity of the entire iteration process is $O(NP \times D \times G)$.

In summary, the overall time complexity of the DPR-MGDE algorithm is $O(NP \times D \times G)$. This shows that the algorithm has a linear time complexity when dealing with large-scale optimisation problems and is suitable for large-scale optimisation tasks. The space complexity of the DPR-MGDE algorithm mainly consists of the following parts:

1. Population storage: The space complexity of storing the positions and fitness values of individuals in the population is $O(NP \times D)$.
2. Historical information storage: The space complexity of storing historical update frequency and other information is $O(NP)$.

In summary, the overall space complexity of the DPR-MGDE algorithm is $O(NP \times D)$. This shows that the algorithm has a linear space complexity when dealing with large-scale optimisation problems and is suitable for large-scale optimisation tasks.

4. Experiments

In this study, we conduct an in-depth experimental analysis of the effectiveness of the DPR-MGDE algorithm and its key components, which include the mutation policy, the collision rate threshold and the restart mechanism. To ensure a high level of confidence in our statistical conclusions, we used the IEEE CEC 2017 test suite, which consists of 30 different benchmark test functions [34] that cover optimisation problems of different dimensions (specifically dimensions 10, 30, 50 and 100). In the experiments, each algorithm was repeated 51 times for each test function to ensure the reliability of the results. We quantified the performance of each algorithm by calculating the Mean Error (Mean Error) and Standard Deviation (Std. Dev). To further verify the significance of the performance differences between algorithms, we applied the Wilcoxon rank sum test [35] statistical test method to evaluate and compare the performance differences between algorithms.

In the experiments, we set the upper limit of the number of function evaluations for each algorithm to 10,000 times multiplied by the problem dimension to ensure that the algorithms have enough chances to converge. In addition, all experiments were conducted on computers configured with MATLAB R2023a to ensure consistency of the experimental environment and reproducibility of the results. Through this series of experimental designs and analyses, we aim to comprehensively evaluate the performance of the DPR-MGDE algorithm and its components and compare it with existing algorithms.

4.1. Experimental Comparison of Variation Strategies

In Table 1, comparing the DE/rand/1 variant using the bootstrap with DE/rand/1, DP-DE/rand/1 outperforms DE/rand/1 on 22 benchmark functions, of which, it clearly outperforms DE/rand/1 on all the hybrid functions (F11–F20), and performs well on the combined functions (F21–F30) and better or flat performance relative to DE/rand/1 on the eight benchmark functions, which shows that the introduction of the bootstrap into the DE/rand/1 mutation strategy is able to improve the algorithm effectiveness.

Table 1. Experimental comparison of improved DP-DE/rand/1 on CEC2017 (D = 30).

	DE/rand/1		**DP-DE/rand/1**		
	Mean	**Std. Dev**	**Mean**	**Std. Dev**	
F1	4.89×10^{-1}	2.97×10^{-1}	$\mathbf{4.51 \times 10^{-1}}$	1.76×10^{-1}	$+$
F2	-	-	-	-	$=$
F3	3.49×10^{4}	6.15×10^{3}	$\mathbf{2.38 \times 10^{4}}$	3.96×10^{3}	$+$
F4	5.57×10^{1}	8.55×10^{0}	6.73×10^{1}	1.06×10^{1}	$-$
F5	1.63×10^{2}	9.15×10^{0}	1.72×10^{2}	1.30×10^{1}	$-$
F6	1.00×10^{-5}	2.00×10^{-5}	1.00×10^{-5}	0.00×10^{0}	$=$
F7	2.19×10^{2}	1.09×10^{1}	$\mathbf{2.11 \times 10^{2}}$	8.67×10^{0}	$+$
F8	1.83×10^{2}	9.08×10^{0}	$\mathbf{1.79 \times 10^{2}}$	7.58×10^{0}	$+$
F9	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}	$=$
F10	6.92×10^{3}	2.57×10^{2}	$\mathbf{6.71 \times 10^{3}}$	2.67×10^{2}	$+$
F11	7.71×10^{1}	1.81×10^{1}	$\mathbf{7.60 \times 10^{1}}$	2.77×10^{1}	$+$
F12	1.82×10^{6}	1.33×10^{6}	$\mathbf{1.29 \times 10^{5}}$	1.65×10^{5}	$+$
F13	6.75×10^{2}	1.40×10^{2}	$\mathbf{4.89 \times 10^{2}}$	8.61×10^{1}	$+$
F14	7.89×10^{1}	6.75×10^{0}	$\mathbf{7.27 \times 10^{1}}$	6.16×10^{0}	$+$
F15	7.16×10^{1}	6.10×10^{0}	$\mathbf{6.12 \times 10^{1}}$	7.73×10^{0}	$+$
F16	1.11×10^{3}	1.93×10^{2}	$\mathbf{9.81 \times 10^{2}}$	1.67×10^{2}	$+$
F17	2.81×10^{2}	6.07×10^{1}	$\mathbf{8.47 \times 10^{1}}$	7.86×10^{0}	$+$
F18	1.08×10^{4}	4.21×10^{3}	$\mathbf{6.92 \times 10^{3}}$	2.21×10^{3}	$+$
F19	2.66×10^{1}	3.46×10^{0}	$\mathbf{2.62 \times 10^{1}}$	2.09×10^{0}	$+$
F20	1.84×10^{2}	1.41×10^{2}	$\mathbf{3.27 \times 10^{1}}$	6.71×10^{0}	$+$
F21	3.59×10^{2}	9.30×10^{0}	$\mathbf{3.32 \times 10^{2}}$	1.03×10^{1}	$+$
F22	1.00×10^{2}	0.00×10^{0}	1.00×10^{2}	0.00×10^{0}	$=$
F23	5.24×10^{2}	9.23×10^{0}	$\mathbf{5.19 \times 10^{2}}$	8.40×10^{0}	$+$
F24	5.88×10^{2}	1.23×10^{1}	$\mathbf{5.86 \times 10^{2}}$	8.00×10^{0}	$+$
F25	3.87×10^{2}	2.57×10^{-2}	3.87×10^{2}	1.19×10^{-2}	$=$
F26	2.65×10^{3}	9.46×10^{1}	2.53×10^{3}	1.10×10^{2}	$+$
F27	4.92×10^{2}	1.05×10^{1}	5.04×10^{2}	8.07×10^{0}	$-$
F28	3.28×10^{2}	4.68×10^{1}	$\mathbf{3.15 \times 10^{2}}$	3.59×10^{1}	$+$
F29	9.08×10^{2}	9.68×10^{1}	$\mathbf{8.25 \times 10^{2}}$	7.37×10^{1}	$+$
F30	1.59×10^{4}	3.19×10^{3}	2.33×10^{4}	1.47×10^{3}	$-$
Total number of (+/=/−): 21/5/4					

Bold data indicates the best results experimentally obtained on a certain test function.

In Table 2, DP-DE/current-to-best/1 performs better than or equal to DE/current-to-best/1 on 20 functions, and on F3, F6, F7, F8, F9, F10, F11, F14, F15, F16, F17, F18, F19, F22, F23, F24, F26, F28 and F30, the mean error of DP-DE/current-to-best/1 is significantly lower than that of DE, indicating that DP-DE is better optimised on these functions. Overall, DP-DE/current-to-best/ is an effective and improved algorithm despite its poor performance on some specific functions.

Table 2. Experimental comparison of improved DP-DE/current-to-best/1 on CEC2017 (D = 30).

	DE/current_to_best/1		**DP-DE/current_to_best/1**		
	Mean	**Std. Dev**	**Mean**	**Std. Dev**	
F1	4.62×10^{3}	1.32×10^{4}	$\mathbf{2.61 \times 10^{3}}$	4.85×10^{3}	$+$
F2	-	-	-	-	$=$
F3	1.57×10^{2}	1.76×10^{2}	$\mathbf{1.38 \times 10^{0}}$	2.05×10^{0}	$+$
F4	$\mathbf{1.00 \times 10^{2}}$	1.39×10^{1}	1.05×10^{2}	1.84×10^{1}	$-$
F5	$\mathbf{1.94 \times 10^{1}}$	1.21×10^{1}	2.72×10^{1}	5.30×10^{0}	$-$

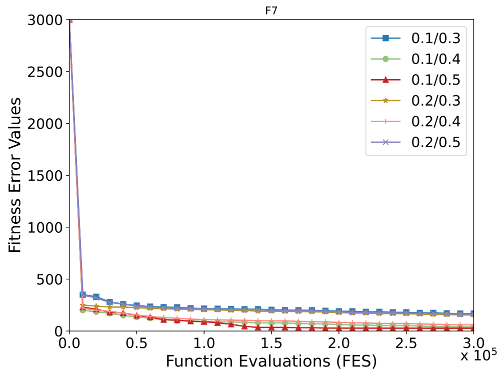
Table 2. *Cont.*

	DE/current_to_best/1		DP-DE/current_to_best/1		
	Mean	**Std. Dev**	**Mean**	**Std. Dev**	
F6	5.26×10^{-3}	4.42×10^{-2}	$\mathbf{2.10 \times 10^{-3}}$	1.15×10^{-2}	+
F7	1.46×10^{2}	1.18×10^{1}	$\mathbf{4.55 \times 10^{1}}$	3.93×10^{1}	+
F8	1.72×10^{2}	1.23×10^{1}	$\mathbf{2.36 \times 10^{1}}$	3.01×10^{1}	+
F9	3.85×10^{-1}	4.79×10^{-1}	$\mathbf{2.97 \times 10^{-1}}$	3.54×10^{-1}	+
F10	5.84×10^{3}	4.80×10^{2}	$\mathbf{4.16 \times 10^{3}}$	1.82×10^{3}	+
F11	6.30×10^{1}	3.11×10^{1}	$\mathbf{4.26 \times 10^{1}}$	2.94×10^{1}	+
F12	$\mathbf{2.43 \times 10^{4}}$	1.35×10^{4}	2.98×10^{4}	1.39×10^{4}	−
F13	$\mathbf{6.56 \times 10^{3}}$	3.54×10^{3}	7.04×10^{3}	7.31×10^{3}	−
F14	1.44×10^{2}	2.39×10^{1}	$\mathbf{5.74 \times 10^{1}}$	2.52×10^{1}	+
F15	2.20×10^{2}	8.37×10^{1}	$\mathbf{1.86 \times 10^{2}}$	8.90×10^{1}	+
F16	7.41×10^{2}	2.38×10^{2}	$\mathbf{4.65 \times 10^{2}}$	2.52×10^{2}	+
F17	1.84×10^{2}	5.30×10^{1}	$\mathbf{9.33 \times 10^{1}}$	5.64×10^{1}	+
F18	5.83×10^{4}	2.69×10^{4}	$\mathbf{5.71 \times 10^{4}}$	3.72×10^{4}	+
F19	$\mathbf{7.06 \times 10^{1}}$	2.97×10^{1}	1.15×10^{2}	5.14×10^{1}	−
F20	$\mathbf{2.28 \times 10^{2}}$	8.02×10^{1}	2.98×10^{2}	8.02×10^{1}	−
F21	$\mathbf{3.32 \times 10^{2}}$	1.03×10^{1}	3.61×10^{2}	1.88×10^{1}	−
F22	2.97×10^{2}	1.08×10^{3}	$\mathbf{1.03 \times 10^{2}}$	1.11×10^{0}	+
F23	4.61×10^{2}	2.18×10^{1}	$\mathbf{3.82 \times 10^{2}}$	1.42×10^{1}	+
F24	5.50×10^{2}	1.67×10^{1}	$\mathbf{4.58 \times 10^{2}}$	1.60×10^{1}	+
F25	3.05×10^{2}	1.72×10^{1}	3.05×10^{2}	1.72×10^{1}	=
F26	1.93×10^{3}	5.07×10^{2}	$\mathbf{1.09 \times 10^{3}}$	2.67×10^{2}	+
F27	$\mathbf{5.11 \times 10^{2}}$	1.17×10^{1}	5.26×10^{2}	1.49×10^{1}	−
F28	4.71×10^{2}	3.44×10^{1}	$\mathbf{4.24 \times 10^{2}}$	3.01×10^{1}	+
F29	7.08×10^{2}	1.07×10^{2}	$\mathbf{5.25 \times 10^{2}}$	8.37×10^{1}	+
F30	9.84×10^{3}	7.43×10^{3}	$\mathbf{7.23 \times 10^{3}}$	3.62×10^{3}	+
Total number of (+/=/−): 20/2/8					

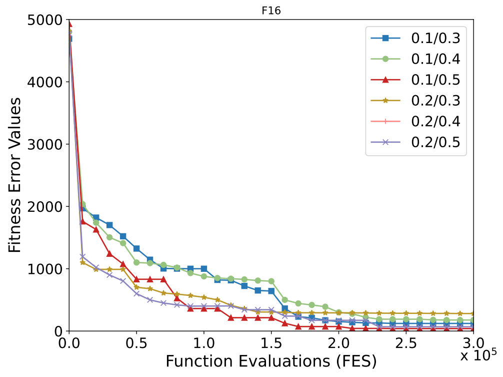
Bold data indicates the best results experimentally obtained on a certain test function.

4.2. Collision Frequency Threshold Parameter Tuning

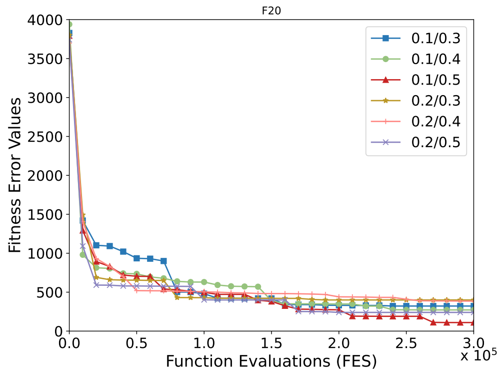
Since this paper uses the collision frequency (determined by the initial/final value according to the formula) as a condition for triggering the restart strategy, we will further calculate the impact of different thresholds on the performance of the algorithm. In this paper, we select four representative benchmark functions in the CEC2017 test set for experiments, single-objective function F7, multi-peak function F16, combined function F20 and F24 and set six groups of different collision frequency threshold parameters 0.1/0.3, 0.1/0.4, 0.1/0.5, 0.2/0.3, 0.2/0.4 and 0.2/0.5. Figure 2 shows the six groups of convergence plots for different parameters for the above four benchmark functions (D = 30). In Figure 2a, the optimal solution obtained by the 0.1/0.5 parameter is the smallest, while the six groups of parameters perform roughly the same in terms of function convergence; in Figure 2b, the 0.2/0.5 parameter has the fastest convergence, the 0.1/0.5 parameter jumps out of the local optimum the most and the 0.1/0.3 and 0.1/0.4 parameters jump out of the local optimum in a good way; in Figure 2c, the initial value of the of 0.2 parameter has the fastest convergence, but the change of fitness is less obvious in the later stage, and the three sets of parameters with initial value of 0.1 have the most obvious effect of jumping out of the local optimum, especially 0.1/0.5; in Figure 2d, the three sets of parameters with initial value of 0.2 basically fail to jump out of the local optimum in the later stage in the fast convergence, and the change of fitness in the first and middle stages of the 0.1/0.5 parameters is most significant.



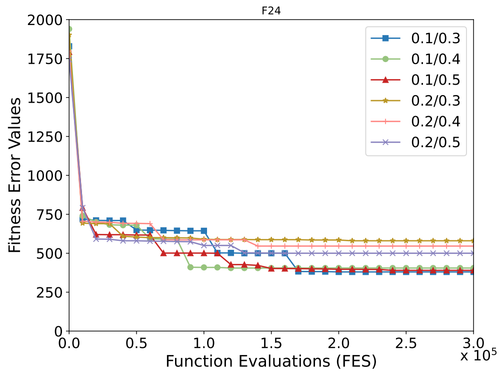
(**a**) F7 convergence plot



(**b**) F16 convergence plot



(**c**) F20 convergence plot



(**d**) F24 convergence plot

Figure 2. Convergence plots for six sets of parameter values on four functions.

In summary, in this paper, the initial and final values of collision frequency are set to 0.1 and 0.5, respectively, and the performance of the 0.1/0.5 parameter group is the best in the above four benchmark functions in a comprehensive way, whether considering the convergence speed or the ability to jump out of the local optimum. This is because, in the early iteration of the differential evolution algorithm, we need the population to maintain good diversity to explore a larger space, so a smaller frequency threshold can trigger restart earlier to achieve the purpose of maintaining the diversity of the population and avoid premature convergence, while in the late iteration, we need the algorithm to have a better ability to develop locally, so a larger frequency threshold can enable the algorithm to focus more on the development. By adjusting the threshold adaptively, the algorithm can better balance the ability of exploration and development.

4.3. Comparison of DPR-MGDE with Current Better Methods

In order to convincingly validate the merits of DPR-MGDE, seven typical DE variants were chosen as opponents with D = 10, D = 30, D = 50 and D = 100. These comparison algorithms are NPADE [36], IDE-EDA [37], FADE [38], MPEDE [39], ADE-DMRM [23], MMDE [21] and DPMADE [5], and their parameter settings are given in detail in Table 3. The numerical and statistical results based on the Wilcoxon rank sum test are shown in Table 2.

Based on the data presented in Table 4, it is evident that the DPR-MGDE algorithm exhibits exceptional performance on the CEC2017 benchmark suite D set to 10. Notably, DPR-MGDE achieves an average ranking of 2.28 across all test functions, which places it at the top among the eight algorithms under comparison, thereby underscoring its superior efficacy. In the domain of single-objective test functions ranging from F1 to F10, IDE-EDA demonstrates the best performance on eight functions, whereas DPR-MGDE secures the highest ranking on three functions. The distribution of rankings for the remaining functions is as follows: F4 (fifth), F5 (fourth), F6 (fourth), F7 (fourth), F8 (fourth) and F10 (second). Regarding the mixed functions from F11 to F20, IDE-EDA attains the best results on F11, F14 and F19; NPADE achieves the best outcomes on F16; FADE is the top performer on F15; and DPR-MGDE ranks first on F18. It is worth highlighting that DPR-MGDE maintains a ranking within the top four for the remaining eight hybrid functions: F11 (third), F14 (fourth), F15 (third), F16 (second), F17 (second), F18 (fourth), F19 (third) and F20 (fourth). Furthermore, in the category of composite functions from F21 to F30, DPR-MGDE ranks first on eight functions, trailing only behind FADE on F22, and is outperformed only by NPADE and itself on F24. When considering the entire spectrum of test problems, DPR-MGDE not only excels in single-objective and hybrid functions, but also demonstrates robust performance in tackling more complex challenges such as composite functions. This comprehensive assessment further substantiates the efficiency and reliability of the DPR-MGDE algorithm in addressing intricate optimisation problems.

Table 3. Parameter settings.

Methods	**Parameter Setting**
NPADE [36]	$NP_{ini} = 8D, NP_{min} = 4, N_{UP-limit} = 20, F_m^0 = Cr_m^0 = 0.5, c = 0.1$
IDE-EDA [37]	$NP_{ini} = 75 \cdot D(2/3), k = 3, H = 5, NP_{min} = 4, r\_arc = 1, s = 0.9$
DADE [38]	$NP_{ini} = 75, NP_{max} = PS_{ini}, NP_{min} = NP_{ini}/3, ss = 3,$ $ns = 25$ or $40, MaxGimp = MaxGstag = 2$
MPEDE [39]	$NP_{ini} = 18D, NP_{min} = 4, p = 0.11, H = 5, m = 2, C = 30$
ADE-DMRM [23]	$\mu_F = \mu_{CR} = 0.5, F{\sim}L(\mu_F, 0.2), CR{\sim}L(\mu_{CR}, 0.1),$ $ps = 18 \cdot D{\sim}10, n = 40, \xi = 0.001, H = 5, p = 0.2{\sim}0.05,$ $r^{rac} = 0.6, r^{rac,B} = 1.6$
MMDE [21]	$M_F^{init} = 0.5, NP = 18 \cdot D{\sim}4, p = 0.11, H = 0.5, r^{arc} = 1.4$
DPMADE [26]	$NP_{ini} = 15D, NP_{min} = 4, LS = 10, p \in (0.1, 0.2), NEP = 10$
DPR-MGDE	$C_{initial} = 0.1, C_{final} = 0.5, F_{min} = 0.4, F_{max} = 1.0,$ $CR_{min} = 0.2, R_{max} = 0.9$

Table 4. Experimental results of DPR-MGDE and 10 famous or up-to-date DE variants on CEC2017 test suite when D = 10.

Func	**Statistic**	**NPADE**	**IDE-EDA**	**FADE**	**MPEDE**	**ADE-DMRM**	**MMDE**	**DPMADE**	**DPR-MGDE**
f1	Mean Error	**0.00×10^{0} (1)**	**0.00×10^{0} (1)**	1.37×10^{-8} **(8)**	**0.00×10^{0} (1)**	**0.00×10^{0} (1)**	**0.00×10^{0} (1)**	**0.00×10^{0} (1)**	**0.00×10^{0} (1)**
	Std Dev	0.00×10^{0}	0.00×10^{0}	1.22×10^{-7}	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}
f2	Mean Error	-	-	-	-	-	-	-	-
	Std Dev	-	-	-	-	-	-	-	-
f3	Mean Error	**0.00×10^{0} (1)**	**0.00×10^{0} (1)**	**0.00×10^{0} (1)**	1.71×10^{-8} (7)	2.09×10^{-15} (8)	**0.00×10^{0} (1)**	**0.00×10^{0} (1)**	**0.00×10^{0} (1)**
	Std Dev	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**4.62×10^{-8}**	**1.04×10^{-14}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**
f4	Mean Error	**0.00×10^{0} (1)**	**0.00×10^{0} (1)**	**0.00×10^{0} (1)**	2.43×10^{1} (6)	7.81×10^{0} (4)	2.88×10^{1} (8)	2.54×10^{1} (7)	2.40×10^{1} (5)
	Std Dev	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}	1.13×10^{1}	2.65×10^{0}	1.28×10^{1}	4.62×10^{0}	3.09×10^{1}
f5	Mean Error	1.62×10^{0} (3)	**1.18×10^{0} (1)**	4.27×10^{0} (7)	2.25×10^{0} (5)	6.14×10^{0} (8)	1.31×10^{0} (2)	2.46×10^{0} (6)	2.11×10^{0} (4)
	Std Dev	9.53×10^{-1}	8.44×10^{-1}	1.54×10^{0}	1.04×10^{0}	2.40×10^{0}	8.09×10^{-1}	9.62×10^{-1}	5.39×10^{0}

Table 4. *Cont.*

Func	Statistic	NPADE	IDE-EDA	FADE	MPEDE	ADE-DMRM	MMDE	DPMADE	DPR-MGDE
f6	Mean Error	1.30×10^{-14}	**0.00×10^{0}**	2.23×10^{-15}	1.34×10^{-14}	4.25×10^{-7}	2.33×10^{-15}	3.12×10^{-14}	1.28×10^{-14}
		(5)	**(1)**	(2)	(6)	(8)	(3)	(7)	(4)
	Std Dev	3.90×10^{-14}	0.00×10^{0}	1.59×10^{-14}	3.70×10^{-14}	2.08×10^{-6}	1.39×10^{-14}	5.12×10^{-14}	2.06×10^{-14}
f7	Mean Error	1.19×10^{1}	1.17×10^{1}	1.55×10^{1}	1.39×10^{1}	1.64×10^{1}	**1.16×10^{1}**	1.20×10^{1}	1.20×10^{1}
		(3)	(2)	(7)	(6)	(8)	**(1)**	(4)	(4)
	Std Dev	7.42×10^{-1}	5.82×10^{-1}	2.28×10^{0}	1.62×10^{0}	2.58×10^{0}	5.14×10^{-1}	6.81×10^{-1}	5.53×10^{-1}
f8	Mean Error	1.70×10^{0}	**1.27×10^{0}**	3.98×10^{0}	2.71×10^{0}	6.53×10^{0}	1.29×10^{0}	2.63×10^{0}	2.11×10^{0}
		(3)	**(1)**	(7)	(6)	(8)	(2)	(5)	(4)
	Std Dev	1.10×10^{0}	7.98×10^{-1}	1.63×10^{0}	1.35×10^{0}	2.40×10^{0}	6.98×10^{-1}	1.40×10^{0}	1.17×10^{0}
f9	Mean Error	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	2.98×10^{-3}	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**
		(1)	**(1)**	**(1)**	**(1)**	(8)	**(1)**	**(1)**	**(1)**
	Std Dev	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}	1.63×10^{-2}	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}
f10	Mean Error	3.59×10^{1}	2.49×10^{1}	1.81×10^{2}	1.04×10^{2}	3.61×10^{2}	**1.04×10^{1}**	1.68×10^{1}	1.27×10^{1}
		(5)	(4)	(7)	(6)	(8)	**(1)**	(3)	(2)
	Std Dev	4.99×10^{1}	5.37×10^{1}	1.11×10^{2}	9.23×10^{1}	2.56×10^{2}	6.23×10^{0}	1.14×10^{1}	9.94×10^{0}
f11	Mean Error	1.20×10^{-1}	**0.00×10^{0}**	4.30×10^{-1}	3.31×10^{-5}	5.64×10^{-1}	9.64×10^{-4}	4.24×10^{-1}	4.52×10^{-5}
		(5)	**(1)**	(7)	(2)	(8)	(4)	(6)	(3)
	Std Dev	3.34×10^{-1}	0.00×10^{0}	6.37×10^{-1}	8.91×10^{-5}	8.13×10^{-1}	5.48×10^{-3}	7.09×10^{-1}	6.36×10^{-5}
f12	Mean Error	2.24×10^{1}	2.73×10^{0}	3.38×10^{2}	4.90×10^{1}	1.52×10^{2}	3.34×10^{0}	7.61×10^{0}	**1.56×10^{0}**
		(2)	(3)	(8)	(6)	(7)	(4)	(5)	**(1)**
	Std Dev	4.55×10^{1}	1.67×10^{1}	1.77×10^{2}	1.17×10^{2}	1.40×10^{2}	1.69×10^{1}	3.70×10^{1}	1.27×10^{1}
f13	Mean Error	3.51×10^{0}	2.43×10^{0}	4.84×10^{0}	6.23×10^{0}	4.08×10^{0}	4.18×10^{0}	4.37×10^{0}	**2.08×10^{0}**
		(3)	(2)	(7)	(8)	(4)	(5)	(6)	**(1)**
	Std Dev	2.67×10^{0}	2.45×10^{0}	2.12×10^{0}	1.08×10^{0}	1.23×10^{0}	4.70×10^{-1}	3.59×10^{-1}	1.63×10^{0}
f14	Mean Error	1.32×10^{0}	**1.95×10^{-2}**	1.48×10^{-1}	4.65×10^{0}	3.53×10^{0}	9.75×10^{-2}	4.72×10^{-1}	3.75×10^{-1}
		(8)	**(1)**	(3)	(6)	(5)	(2)	(7)	(4)
	Std Dev	1.23×10^{0}	1.39×10^{-1}	3.58×10^{-1}	3.06×10^{0}	6.12×10^{0}	2.99×10^{-1}	7.33×10^{-1}	4.11×10^{-1}
f15	Mean Error	1.92×10^{-1}	2.28×10^{-1}	**1.68×10^{-1}**	6.09×10^{-1}	5.19×10^{-1}	3.71×10^{-1}	2.45×10^{-1}	2.03×10^{-1}
		(2)	(4)	**(1)**	(8)	(7)	(6)	(5)	(3)
	Std Dev	2.06×10^{-1}	2.15×10^{-1}	1.67×10^{-1}	7.43×10^{-1}	7.42×10^{-1}	1.91×10^{-1}	2.12×10^{-1}	1.89×10^{-1}
f16	Mean Error	**1.91×10^{-1}**	6.61×10^{-1}	4.28×10^{-1}	1.03×10^{0}	9.54×10^{0}	6.25×10^{-1}	4.64×10^{-1}	3.84×10^{-1}
		(1)	(6)	(3)	(7)	(8)	(5)	(4)	(2)
	Std Dev	1.66×10^{-1}	2.53×10^{-1}	2.54×10^{-1}	2.82×10^{-1}	2.99×10^{1}	5.00×10^{-1}	2.40×10^{-1}	1.96×10^{-1}
f17	Mean Error	1.57×10^{-1}	7.40×10^{-1}	**1.19×10^{0}**	2.00×10^{1}	2.07×10^{1}	2.06×10^{1}	1.61×10^{1}	1.42×10^{1}
		(3)	(5)	**(1)**	(6)	(8)	(7)	(4)	(2)
	Std Dev	2.65×10^{-1}	4.72×10^{-1}	7.79×10^{-1}	7.15×10^{0}	8.34×10^{0}	4.30×10^{0}	7.56×10^{0}	6.81×10^{0}
f18	Mean Error	2.85×10^{-1}	2.80×10^{-1}	3.84×10^{-1}	2.12×10^{0}	9.22×10^{0}	3.64×10^{-1}	**2.77×10^{-1}**	2.89×10^{-1}
		(3)	(2)	(7)	(6)	(8)	(5)	**(1)**	(4)
	Std Dev	2.96×10^{-1}	2.12×10^{-1}	2.43×10^{-1}	5.08×10^{0}	1.01×10^{1}	1.87×10^{-1}	2.01×10^{-1}	2.54×10^{-1}
f19	Mean Error	2.26×10^{-2}	**7.70×10^{-3}**	2.63×10^{-2}	3.34×10^{-1}	2.98×10^{-1}	1.72×10^{-2}	3.66×10^{-2}	1.93×10^{-2}
		(4)	**(1)**	(5)	(8)	(7)	(2)	(6)	(3)
	Std Dev	2.72×10^{-2}	1.10×10^{-2}	2.11×10^{-2}	1.91×10^{-1}	4.20×10^{-1}	1.15×10^{-2}	1.45×10^{-1}	1.22×10^{-2}
f20	Mean Error	**0.00×10^{0}**	5.10×10^{-1}	4.61×10^{-1}	6.80×10^{0}	6.00×10^{0}	7.38×10^{0}	3.17×10^{0}	1.58×10^{0}
		(1)	(3)	(2)	(7)	(6)	(8)	(5)	(4)
	Std Dev	0.00×10^{0}	2.65×10^{-1}	5.02×10^{-1}	2.58×10^{0}	8.19×10^{0}	8.66×10^{0}	2.92×10^{0}	1.13×10^{0}
f21	Mean Error	1.22×10^{2}	1.48×10^{2}	1.13×10^{2}	**1.00×10^{2}**	**1.00×10^{2}**	**1.00×10^{2}**	**1.00×10^{2}**	**1.00×10^{2}**
		(7)	(8)	(6)	**(1)**	**(1)**	**(1)**	**(1)**	**(1)**
	Std Dev	4.28×10^{1}	5.18×10^{1}	3.46×10^{1}	0.00×10^{0}	1.57×10^{-13}	0.00×10^{0}	2.05×10^{-13}	0.00×10^{0}
f22	Mean Error	1.00×10^{2}	1.00×10^{2}	**6.86×10^{1}**	9.61×10^{1}	1.00×10^{2}	9.44×10^{1}	9.43×10^{1}	7.25×10^{1}
		(6)	(6)	**(1)**	(5)	(6)	(4)	(3)	(2)
	Std Dev	2.26×10^{-13}	5.67×10^{-2}	4.65×10^{1}	1.96×10^{1}	1.39×10^{-13}	2.25×10^{1}	2.31×10^{1}	3.81×10^{1}
f23	Mean Error	3.02×10^{2}	3.00×10^{2}	3.04×10^{2}	3.45×10^{2}	3.53×10^{2}	3.45×10^{2}	**2.00×10^{2}**	**2.00×10^{2}**
		(4)	(3)	(5)	(6)	(8)	(6)	**(1)**	**(1)**
	Std Dev	1.90×10^{0}	8.56×10^{-1}	2.24×10^{0}	2.24×10^{0}	3.81×10^{0}	2.94×10^{0}	0.00×10^{0}	0.00×10^{0}

Table 4. *Cont.*

Func	Statistic	NPADE	IDE-DAE	FADE	MPEDE	ADE-DMRM	MMDE	DPMADE	DPR-MGDE
f24	Mean Error	**1.75×10^2 (1)**	2.82×10^2 (5)	2.55×10^2 (4)	2.98×10^2 (6)	3.00×10^2 (7)	3.72×10^2 (8)	2.00×10^2 (2)	2.33×10^2 (3)
	Std Dev	1.08×10^2	9.69×10^1	1.11×10^2	5.46×10^1	1.09×10^2	7.52×10^1	0.00×10^0	1.64×10^0
f25	Mean Error	4.03×10^2 (4)	4.14×10^2 (6)	4.01×10^2 (2)	4.17×10^2 (7)	4.24×10^2 (8)	4.01×10^2 (2)	4.12×10^2 (5)	**2.00×10^2 (1)**
	Std Dev	1.48×10^1	2.19×10^1	1.10×10^1	2.34×10^1	2.31×10^1	1.70×10^1	2.57×10^1	0.00×10^0
f26	Mean Error	3.00×10^2 (4)	3.00×10^2 (4)	3.00×10^2 (4)	2.98×10^2 (3)	3.85×10^2 (8)	3.00×10^2 (4)	2.30×10^2 (2)	**1.97×10^2 (1)**
	Std Dev	0.00×10^0	0.00×10^0	0.00×10^0	1.40×10^1	1.65×10^2	1.11×10^{-13}	2.06×10^0	1.18×10^1
f27	Mean Error	3.89×10^2 (3)	3.89×10^2 (3)	3.89×10^2 (3)	4.05×10^2 (6)	4.29×10^2 (8)	4.21×10^2 (7)	2.09×10^2 (2)	**2.00×10^2 (1)**
	Std Dev	9.13×10^{-1}	1.54×10^{-1}	1.73×10^0	1.03×10^1	1.29×10^1	1.16×10^1	4.27×10^1	0.00×10^0
f28	Mean Error	3.22×10^2 (6)	3.06×10^2 (4)	3.00×10^2 (2)	3.16×10^2 (5)	3.31×10^2 (7)	3.03×10^2 (3)	3.43×10^2 (8)	**2.02×10^2 (1)**
	Std Dev	7.70×10^1	3.97×10^1	0.00×10^0	2.33×10^1	2.40×10^1	1.16×10^1	1.77×10^1	1.28×10^1
f29	Mean Error	2.35×10^2 (4)	2.35×10^2 (4)	2.44×10^2 (8)	2.42×10^2 (7)	2.39×10^2 (6)	2.31×10^2 (3)	2.16×10^2 (2)	**2.00×10^2 (1)**
	Std Dev	4.92×10^0	3.68×10^0	5.14×10^0	6.46×10^0	8.03×10^0	3.21×10^0	1.82×10^1	0.00×10^0
f30	Mean Error	4.30×10^2 (6)	1.64×10^4 (8)	4.42×10^2 (7)	2.88×10^2 (5)	2.28×10^2 (3)	2.86×10^2 (4)	2.17×10^2 (2)	**2.00×10^2 (1)**
	Std Dev	1.70×10^2	1.14×10^5	1.90×10^1	1.54×10^2	1.04×10^1	3.72×10^1	1.88×10^0	0.00×10^0
	Rank	3.38	3.17	4.38	5.49	6.59	3.79	3.86	2.28

Bold data indicates the best results experimentally obtained on a certain test function.

Based on the data presented in Table 5, it is evident that the DPR-MGDE algorithm achieves the best results among all comparison functions when D is set to 30, with an average ranking of 2.24. This ranking underscores the algorithm's superior performance across a diverse set of test functions. In the category of single-objective functions (F1–F10), DPR-MGDE attains the optimal solutions in F1, F7 and F9, outperforming all other comparison functions. The rankings for the remaining benchmark functions are as follows: F3 (fourth), F4–F6 (second), F8 (second) and F10 (sixth). These results highlight DPR-MGDE's consistent performance in single-objective optimisation problems. In the hybrid functions (F11–F20), DPR-MGDE demonstrates robust performance, achieving the best result on one benchmark function (F14). Moreover, it outperforms more than half of the comparison functions on all other functions, except for F11. The detailed rankings are as follows: F11 (fifth), F12 (fourth), F13 (fourth), F14 (first), F15 (third), F16 (second), F17 (third), F18 (third), F19 (fourth) and F20 (fourth). This distribution of rankings indicates DPR-MGDE's ability to handle complex hybrid functions effectively. In the combined functions (F21–F30), DPR-MGDE exhibits exceptionally superior performance, ranking as the best performer on nine benchmark functions. The only exception is F29, where it trails behind ADE-DMRM. This performance highlights DPR-MGDE's capability to tackle highly complex optimisation problems, further validating its efficiency and reliability in a wide range of applications. Overall, the comprehensive evaluation of DPR-MGDE across different types of functions and dimensions confirms its robustness and effectiveness in solving challenging optimisation problems, making it a valuable addition to the field of optimisation algorithms.

Table 5. Experimental results of DPR-MGDE and 10 famous or up-to-date DE variants on CEC2017 test suite when D = 30.

Func	Statistic	NPADE	IDE-EDA	FADE	MPEDE	ADE-DMRM	MMDE	DPMADE	DPR-MGDE
f1	Mean Error	2.09×10^{-16} (5)	**0.00×10^{0} (1)**	1.66×10^{0} (8)	3.47×10^{-14} (7)	**0.00×10^{0} (1)**	2.19×10^{-16} (6)	**0.00×10^{0} (1)**	**0.00×10^{0} (1)**
	Std Dev	1.99×10^{-15}	0.00×10^{0}	8.12×10^{0}	1.74×10^{-14}	0.00×10^{0}	1.99×10^{-15}	0.00×10^{0}	0.00×10^{0}
f2	Mean Error	-	-	-	-	-	-	-	-
	Std Dev	-	-	-	-	-	-	-	-
f3	Mean Error	6.67×10^{2} (8)	1.51×10^{-15} (3)	2.33×10^{-14} (6)	4.06×10^{-13} (7)	**0.00×10^{0} (1)**	2.01×10^{-14} (5)	**0.00×10^{0} (1)**	5.20×10^{-15} (4)
	Std Dev	1.23×10^{3}	7.96×10^{-15}	2.45×10^{-14}	2.22×10^{-13}	0.00×10^{0}	2.74×10^{-14}	0.00×10^{0}	1.98×10^{-14}
f4	Mean Error	4.03×10^{1} (7)	5.77×10^{1} (8)	2.70×10^{1} (6)	8.80×10^{-12} (5)	**3.57×10^{-14} (1)**	5.33×10^{-14} (3)	5.86×10^{-14} (4)	4.91×10^{-14} (2)
	Std Dev	8.44×10^{0}	7.78×10^{-1}	3.34×10^{1}	1.69×10^{-11}	2.78×10^{-14}	1.35×10^{-14}	1.39×10^{-14}	2.85×10^{-14}
f5	Mean Error	2.31×10^{1} (6)	7.40×10^{0} (4)	3.25×10^{1} (7)	3.41×10^{1} (8)	**6.22×10^{0} (1)**	7.01×10^{0} (3)	2.21×10^{1} (5)	6.71×10^{0} (2)
	Std Dev	6.44×10^{0}	2.09×10^{0}	9.02×10^{0}	1.10×10^{1}	1.50×10^{0}	1.59×10^{0}	5.11×10^{0}	1.33×10^{0}
f6	Mean Error	3.67×10^{-8} (6)	4.03×10^{-9} (4)	1.63×10^{-3} (7)	1.76×10^{-1} (8)	8.83×10^{-9} (5)	3.44×10^{-9} (3)	**2.01×10^{-14} (1)**	2.68×10^{-9} (2)
	Std Dev	1.94×10^{-7}	2.01×10^{-8}	6.91×10^{-3}	1.40×10^{-1}	3.29×10^{-8}	1.97×10^{-8}	4.38×10^{-14}	1.92×10^{-8}
f7	Mean Error	4.06×10^{1} (6)	3.22×10^{1} (2)	5.94×10^{1} (8)	5.81×10^{1} (7)	3.77×10^{1} (4)	3.75×10^{1} (3)	3.88×10^{1} (5)	**2.74×10^{1} (1)**
	Std Dev	5.92×10^{0}	1.56×10^{0}	6.50×10^{0}	1.04×10^{1}	1.10×10^{0}	1.51×10^{0}	8.03×10^{-1}	1.49×10^{0}
f8	Mean Error	2.18×10^{1} (6)	7.60×10^{0} (5)	3.32×10^{1} (8)	3.02×10^{1} (7)	7.00×10^{0} (3)	7.04×10^{0} (4)	**3.15×10^{0} (1)**	6.92×10^{0} (2)
	Std Dev	3.59×10^{0}	2.07×10^{0}	1.04×10^{1}	1.07×10^{1}	1.47×10^{0}	1.78×10^{0}	1.53×10^{0}	1.19×10^{0}
f9	Mean Error	2.98×10^{-12} (6)	**0.00×10^{0} (1)**	3.18×10^{-1} (7)	7.66×10^{1} (8)	**0.00×10^{0} (1)**	**0.00×10^{0} (1)**	**0.00×10^{0} (1)**	**0.00×10^{0} (1)**
	Std Dev	4.98×10^{-11}	0.00×10^{0}	5.28×10^{-1}	6.70×10^{1}	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}
f10	Mean Error	**3.32×10^{2} (1)**	1.53×10^{3} (5)	2.44×10^{3} (7)	2.82×10^{3} (8)	1.16×10^{3} (3)	1.27×10^{3} (4)	1.09×10^{3} (2)	1.58×10^{3} (6)
	Std Dev	4.30×10^{2}	2.92×10^{2}	6.44×10^{2}	6.63×10^{2}	1.63×10^{2}	1.82×10^{2}	1.91×10^{2}	1.84×10^{2}
f11	Mean Error	1.51×10^{1} (4)	**3.77×10^{0} (1)**	1.97×10^{1} (7)	1.13×10^{2} (8)	9.12×10^{0} (2)	9.01×10^{0} (3)	1.78×10^{1} (6)	1.72×10^{1} (5)
	Std Dev	5.63×10^{0}	1.16×10^{1}	1.73×10^{1}	4.63×10^{1}	3.87×10^{0}	1.88×10^{0}	2.47×10^{1}	3.52×10^{0}
f12	Mean Error	7.98×10^{2} (5)	**1.46×10^{2} (1)**	5.12×10^{3} (8)	1.57×10^{3} (7)	1.22×10^{3} (6)	2.16×10^{2} (2)	5.28×10^{2} (3)	6.17×10^{2} (4)
	Std Dev	3.30×10^{2}	9.87×10^{1}	6.73×10^{3}	4.49×10^{2}	3.62×10^{2}	1.58×10^{2}	2.94×10^{2}	3.60×10^{2}
f13	Mean Error	1.53×10^{1} (3)	**1.29×10^{1} (1)**	1.34×10^{2} (8)	7.46×10^{1} (7)	1.79×10^{1} (5)	2.02×10^{1} (6)	1.49×10^{1} (2)	1.68×10^{1} (4)
	Std Dev	6.98×10^{0}	7.11×10^{0}	6.93×10^{1}	5.60×10^{1}	4.52×10^{0}	7.16×10^{-1}	4.23×10^{0}	3.02×10^{0}
f14	Mean Error	2.60×10^{1} (7)	2.12×10^{1} (3)	2.30×10^{1} (5)	5.83×10^{1} (8)	2.24×10^{1} (4)	2.11×10^{1} (2)	2.32×10^{1} (6)	**2.05×10^{1} (1)**
	Std Dev	4.94×10^{0}	9.03×10^{-1}	1.00×10^{1}	1.95×10^{1}	1.42×10^{0}	9.53×10^{-1}	1.57×10^{0}	6.97×10^{0}
f15	Mean Error	7.83×10^{0} (6)	**6.05×10^{-1} (1)**	1.66×10^{1} (7)	5.10×10^{1} (8)	4.79×10^{0} (4)	8.10×10^{-1} (2)	5.19×10^{0} (5)	3.04×10^{0} (3)
	Std Dev	4.73×10^{0}	5.66×10^{-1}	9.36×10^{0}	2.85×10^{1}	2.67×10^{0}	5.82×10^{-1}	2.74×10^{0}	1.88×10^{0}
f16	Mean Error	1.49×10^{2} (7)	**2.11×10^{1} (1)**	4.98×10^{2} (8)	4.41×10^{2} (5)	4.23×10^{1} (4)	4.66×10^{1} (6)	4.05×10^{1} (3)	2.94×10^{1} (2)
	Std Dev	1.31×10^{2}	3.39×10^{1}	2.30×10^{2}	2.29×10^{2}	4.31×10^{1}	4.78×10^{1}	5.72×10^{1}	3.85×10^{1}
f17	Mean Error	**2.93×10^{1} (1)**	3.54×10^{1} (2)	7.62×10^{1} (7)	1.17×10^{2} (8)	4.98×10^{1} (4)	5.46×10^{1} (6)	4.98×10^{1} (4)	4.89×10^{1} (3)
	Std Dev	1.11×10^{1}	5.99×10^{0}	7.73×10^{1}	5.40×10^{1}	9.20×10^{0}	9.27×10^{0}	7.88×10^{0}	8.51×10^{0}

Table 5. *Cont.*

Func	Statistic	NPADE	IDE-EDA	FADE	MPEDE	ADE-DMRM	MMDE	DPMADE	DPR-MGDE
f18	Mean Error	2.29×10^{1} (4)	2.08×10^{1} (2)	3.68×10^{1} (7)	1.59×10^{2} (8)	2.61×10^{1} (6)	$\mathbf{1.93 \times 10^{1}}$ **(1)**	2.31×10^{1} (5)	2.23×10^{1} (3)
	Std Dev	1.60×10^{0}	3.36×10^{-1}	1.35×10^{1}	7.35×10^{1}	5.62×10^{0}	6.13×10^{0}	1.62×10^{0}	1.37×10^{0}
f19	Mean Error	8.49×10^{0} (6)	$\mathbf{3.17 \times 10^{0}}$ **(1)**	3.44×10^{1} (7)	5.85×10^{1} (8)	7.50×10^{0} (5)	4.00×10^{0} (2)	4.03×10^{0} (3)	6.09×10^{0} (4)
	Std Dev	1.95×10^{0}	7.94×10^{-1}	2.77×10^{1}	3.74×10^{1}	2.00×10^{0}	1.19×10^{0}	2.12×10^{0}	1.65×10^{0}
f20	Mean Error	$\mathbf{5.11 \times 10^{1}}$ **(1)**	3.05×10^{1} (2)	9.80×10^{1} (3)	1.72×10^{2} (8)	7.80×10^{1} (7)	7.18×10^{1} (5)	7.30×10^{1} (6)	7.05×10^{1} (4)
	Std Dev	5.41×10^{1}	7.02×10^{0}	7.70×10^{1}	1.27×10^{2}	1.10×10^{1}	1.28×10^{1}	4.59×10^{1}	1.13×10^{1}
f21	Mean Error	2.43×10^{2} (8)	2.18×10^{2} (6)	2.30×10^{2} (7)	8.69×10^{0} (2)	1.50×10^{2} (4)	1.50×10^{2} (4)	1.08×10^{2} (3)	$\mathbf{3.30 \times 10^{1}}$ **(1)**
	Std Dev	6.72×10^{0}	2.22×10^{0}	9.68×10^{0}	2.66×10^{1}	2.01×10^{-13}	2.01×10^{-13}	4.12×10^{0}	2.05×10^{1}
f22	Mean Error	1.00×10^{2} (6)	1.00×10^{2} (6)	1.01×10^{2} (8)	3.46×10^{1} (5)	1.50×10^{1} (3)	1.50×10^{1} (3)	1.09×10^{1} (2)	$\mathbf{3.49 \times 10^{0}}$ **(1)**
	Std Dev	6.39×10^{-14}	1.00×10^{-13}	1.75×10^{0}	8.89×10^{0}	2.01×10^{-13}	1.84×10^{-13}	1.82×10^{-13}	1.26×10^{0}
f23	Mean Error	3.69×10^{2} (4)	3.97×10^{2} (7)	3.92×10^{2} (6)	5.30×10^{2} (8)	3.82×10^{2} (5)	2.92×10^{2} (3)	2.55×10^{2} (2)	$\mathbf{2.00 \times 10^{2}}$ **(1)**
	Std Dev	8.81×10^{0}	3.30×10^{0}	1.29×10^{1}	3.01×10^{1}	5.12×10^{0}	7.83×10^{0}	2.72×10^{0}	0.00×10^{0}
f24	Mean Error	4.42×10^{2} (6)	4.26×10^{2} (4)	4.58×10^{2} (7)	3.15×10^{2} (2)	9.46×10^{2} (8)	3.62×10^{2} (3)	4.29×10^{2} (5)	$\mathbf{2.97 \times 10^{2}}$ **(1)**
	Std Dev	7.64×10^{0}	2.22×10^{0}	1.36×10^{1}	1.66×10^{2}	1.45×10^{1}	1.90×10^{2}	3.40×10^{0}	4.08×10^{1}
f25	Mean Error	4.07×10^{2} (6)	3.87×10^{2} (2)	3.87×10^{2} (2)	4.22×10^{2} (8)	4.09×10^{2} (7)	4.00×10^{2} (5)	3.87×10^{2} (2)	$\mathbf{2.00 \times 10^{2}}$ **(1)**
	Std Dev	5.43×10^{-2}	6.46×10^{-3}	1.07×10^{0}	3.01×10^{1}	6.78×10^{0}	9.66×10^{-2}	1.09×10^{-2}	1.60×10^{-2}
f26	Mean Error	1.08×10^{3} (4)	8.96×10^{2} (2)	1.32×10^{3} (6)	1.22×10^{3} (5)	1.88×10^{3} (8)	1.83×10^{3} (7)	9.59×10^{2} (3)	$\mathbf{2.10 \times 10^{2}}$ **(1)**
	Std Dev	9.74×10^{1}	3.28×10^{1}	2.23×10^{2}	7.22×10^{2}	2.93×10^{1}	8.41×10^{1}	5.18×10^{1}	4.63×10^{1}
f27	Mean Error	5.00×10^{2} (4)	4.94×10^{2} (2)	4.99×10^{2} (3)	5.26×10^{2} (6)	7.32×10^{2} (7)	7.37×10^{2} (8)	5.06×10^{2} (5)	$\mathbf{2.63 \times 10^{2}}$ **(1)**
	Std Dev	5.62×10^{0}	7.95×10^{0}	1.07×10^{1}	1.00×10^{2}	8.70×10^{0}	3.35×10^{1}	4.99×10^{0}	1.74×10^{2}
f28	Mean Error	3.17×10^{2} (5)	3.11×10^{2} (2)	3.16×10^{2} (4)	4.67×10^{2} (8)	3.96×10^{2} (7)	3.87×10^{2} (6)	3.12×10^{2} (3)	$\mathbf{2.09 \times 10^{2}}$ **(1)**
	Std Dev	4.00×10^{1}	3.36×10^{1}	4.11×10^{1}	2.84×10^{1}	2.87×10^{1}	3.30×10^{1}	5.09×10^{1}	1.82×10^{1}
f29	Mean Error	4.19×10^{2} (5)	4.47×10^{2} (7)	4.36×10^{2} (6)	3.77×10^{2} (4)	$\mathbf{3.16 \times 10^{2}}$ **(1)**	3.19×10^{2} (2)	4.46×10^{2} (8)	3.19×10^{2} (2)
	Std Dev	1.01×10^{1}	1.84×10^{1}	2.66×10^{1}	8.50×10^{1}	9.56×10^{0}	8.64×10^{0}	1.60×10^{1}	2.05×10^{1}
f30	Mean Error	1.99×10^{3} (7)	1.97×10^{3} (6)	2.03×10^{3} (8)	4.03×10^{2} (3)	9.24×10^{2} (5)	9.17×10^{2} (4)	3.07×10^{2} (2)	$\mathbf{2.94 \times 10^{2}}$ **(1)**
	Std Dev	3.67×10^{1}	1.26×10^{1}	8.91×10^{1}	3.32×10^{2}	5.07×10^{1}	6.27×10^{1}	6.85×10^{1}	2.11×10^{1}
	b/w/s	25/5/0	17/11/2	29/1/0	30/0/0	23/5/2	23/6/1	21/7/2	-
	Rank	5.17	3.24	6.48	6.24	4.17	3.86	3.41	2.24

Bold data indicates the best results experimentally obtained on a certain test function.

Based on the data presented in Table 6, it is observed that the DPR-MGDE algorithm achieves the highest average ranking of 2.66 when D is set to 50, with DPMADE closely following at an average ranking of 2.80. This indicates a highly competitive performance between these two algorithms in high-dimensional optimisation problems. In the category of single-objective functions (F1–F10), the optimal solutions obtained by DPR-MGDE are generally outperformed by other comparison algorithms, suggesting that DPR-MGDE may face challenges in this specific domain of optimisation problems. However, in the hybrid functions (F11–F20), DPR-MGDE demonstrates a significant improvement. It outperforms all other compared algorithms on F13, and surpasses half of the compared algorithms on the remaining functions. The detailed rankings are as follows: F11 (fourth), F12 (fourth), F13 (first), F14 (third), F15 (third), F16 (second), F17 (fourth), F18 (second), F19 (third) and F20 (fifth). These results highlight DPR-MGDE's ability to effectively handle the complexity and multimodality of hybrid functions. In the combined functions (F21–F30), DPR-MGDE shows a remarkable performance, achieving the best optimal solutions on F22, F23, F24, F25, F26, F28, F29 and F30. However, it is noted that DPR-MGDE is less effective compared to ADE-DMRM and DPMADE on F21, and to DPMADE on F27. This suggests that, while DPR-MGDE excels in many complex optimisation scenarios, there are specific instances wherein other algorithms may offer superior performance. Overall, the results from Table 6 provide a nuanced view of DPR-MGDE's performance. While it may not dominate in single-objective functions, its strong performance in hybrid and combined functions, particularly in high-dimensional spaces, confirms its robustness and adaptability in tackling a wide range of optimisation challenges. This comprehensive evaluation further underscores the algorithm's potential as a valuable tool in the optimisation toolkit, especially for problems characterised by high dimensionality and complexity.

Table 6. Experimental results of DPR-MGDE and 10 famous or up-to-date DE variants on CEC2017 test suite when D = 50.

Func	Statistic	NPADE	IDE-DAE	FADE	MPEDE	ADE-DMRM	MMDE	DPMADE	DPR-MGDE
f1	Mean Error	1.44×10^{-12} (3)	1.25×10^{-14} (2)	2.11×10^{2} (8)	1.50×10^{-14} (6)	2.40×10^{-14} (7)	$\mathbf{0.00 \times 10^{0}}$ **(1)**	1.45×10^{-14} (4)	1.48×10^{-14} (5)
	Std Dev	1.43×10^{-12}	4.62×10^{-15}	2.33×10^{2}	3.38×10^{-15}	6.66×10^{-15}	0.00×10^{0}	4.49×10^{-15}	1.07×10^{-15}
f2	Mean Error	-	-	-	-	-	-	-	-
	Std Dev	-	-	-	-	-	-	-	-
f3	Mean Error	1.29×10^{4} (8)	8.36×10^{-14} (3)	5.82×10^{-3} (7)	1.07×10^{-13} (5)	1.72×10^{-13} (6)	$\mathbf{0.00 \times 10^{0}}$ **(1)**	1.06×10^{-13} (4)	4.26×10^{-14} (2)
	Std Dev	1.24×10^{4}	2.87×10^{-14}	1.51×10^{-2}	3.14×10^{-14}	4.89×10^{-14}	0.00×10^{0}	3.77×10^{-14}	2.01×10^{-14}
f4	Mean Error	6.09×10^{1} (7)	4.42×10^{1} (4)	4.90×10^{1} (6)	4.84×10^{1} (5)	6.37×10^{1} (8)	3.65×10^{1} (2)	$\mathbf{3.50 \times 10^{1}}$ **(1)**	3.87×10^{1} (3)
	Std Dev	4.16×10^{1}	3.52×10^{1}	4.39×10^{1}	4.92×10^{1}	4.71×10^{1}	3.94×10^{1}	4.71×10^{1}	3.94×10^{1}
f5	Mean Error	5.18×10^{1} (6)	1.56×10^{1} (4)	7.13×10^{1} (8)	1.42×10^{1} (2)	1.56×10^{1} (4)	5.99×10^{0} (7)	$\mathbf{1.35 \times 10^{1}}$ **(1)**	1.49×10^{1} (3)
	Std Dev	1.04×10^{1}	3.20×10^{0}	1.99×10^{1}	2.46×10^{0}	2.97×10^{0}	1.81×10^{0}	2.00×10^{0}	2.77×10^{0}
f6	Mean Error	6.46×10^{-5} (8)	2.07×10^{-7} (4)	6.83×10^{-7} (6)	1.13×10^{-7} (2)	7.47×10^{-7} (7)	$\mathbf{1.14 \times 10^{-13}}$ **(1)**	2.06×10^{-7} (3)	2.10×10^{-7} (5)
	Std Dev	2.58×10^{-4}	3.47×10^{-7}	1.26×10^{-6}	1.93×10^{-7}	1.16×10^{-6}	0.00×10^{0}	4.64×10^{-7}	3.84×10^{-7}
f7	Mean Error	9.28×10^{1} (7)	7.05×10^{1} (6)	1.62×10^{2} (8)	6.39×10^{1} (4)	6.53×10^{1} (5)	$\mathbf{5.67 \times 10^{1}}$ **(1)**	6.32×10^{1} (3)	6.02×10^{1} (2)
	Std Dev	1.08×10^{1}	3.57×10^{0}	5.67×10^{1}	2.31×10^{0}	2.19×10^{0}	1.21×10^{0}	1.79×10^{0}	2.44×10^{0}
f8	Mean Error	5.21×10^{1} (7)	1.56×10^{1} (5)	7.29×10^{1} (8)	1.32×10^{1} (2)	1.43×10^{1} (4)	5.17×10^{0} (6)	$\mathbf{1.30 \times 10^{1}}$ **(1)**	1.41×10^{1} (3)
	Std Dev	1.12×10^{1}	3.95×10^{0}	2.34×10^{1}	1.88×10^{0}	2.54×10^{0}	2.40×10^{0}	2.03×10^{0}	1.03×10^{1}
f9	Mean Error	1.05×10^{-2} (5)	0.00×10^{0} (1)	3.34×10^{0} (8)	2.90×10^{-14} (3)	9.14×10^{-14} (4)	$\mathbf{0.00 \times 10^{0}}$ **(1)**	2.31×10^{-2} (7)	1.83×10^{-2} (6)
	Std Dev	2.91×10^{-2}	0.00×10^{0}	1.44×10^{1}	5.00×10^{-14}	4.56×10^{-14}	0.00×10^{0}	9.05×10^{-2}	5.11×10^{-2}
f10	Mean Error	3.34×10^{3} (5)	3.96×10^{3} (7)	8.49×10^{3} (8)	3.15×10^{3} (2)	3.20×10^{3} (3)	3.58×10^{3} (6)	$\mathbf{3.03 \times 10^{3}}$ **(1)**	3.27×10^{3} (4)
	Std Dev	7.31×10^{2}	3.24×10^{2}	5.92×10^{2}	2.81×10^{2}	2.90×10^{2}	6.37×10^{2}	3.59×10^{2}	2.09×10^{2}
f11	Mean Error	4.98×10^{1} (7)	$\mathbf{2.41 \times 10^{1}}$ **(1)**	4.89×10^{1} (6)	5.19×10^{1} (8)	3.02×10^{1} (2)	3.18×10^{1} (3)	4.63×10^{1} (5)	3.55×10^{1} (4)
	Std Dev	1.00×10^{1}	3.08×10^{0}	9.97×10^{0}	1.08×10^{1}	9.12×10^{0}	5.04×10^{0}	1.17×10^{1}	1.14×10^{1}

Table 6. *Cont.*

Func	Statistic	NPADE	IDE-DAE	FADE	MPEDE	ADE-DMRM	MMDE	DPMADE	DPR-MGDE
f12	Mean Error	2.40×10^{3} (7)	**1.48×10^{3} (1)**	3.64×10^{4} (8)	2.33×10^{3} (6)	2.00×10^{3} (3)	1.63×10^{3} (2)	2.28×10^{3} (5)	2.21×10^{3} (4)
	Std Dev	4.98×10^{2}	4.75×10^{2}	3.01×10^{4}	4.61×10^{2}	4.43×10^{2}	4.72×10^{2}	5.15×10^{2}	4.63×10^{2}
f13	Mean Error	7.23×10^{1} (7)	3.24×10^{1} (2)	5.87×10^{2} (8)	6.92×10^{1} (6)	5.00×10^{1} (4)	3.76×10^{1} (3)	6.39×10^{1} (5)	**3.07×10^{1} (1)**
	Std Dev	3.09×10^{1}	2.01×10^{1}	5.58×10^{2}	3.11×10^{1}	2.84×10^{1}	1.86×10^{1}	3.33×10^{1}	1.82×10^{1}
f14	Mean Error	3.88×10^{1} (7)	2.34×10^{1} (2)	4.33×10^{1} (8)	3.26×10^{1} (6)	**2.21×10^{1} (1)**	2.89×10^{1} (5)	2.64×10^{1} (4)	2.45×10^{1} (3)
	Std Dev	7.22×10^{0}	1.79×10^{0}	1.16×10^{1}	4.42×10^{0}	1.37×10^{0}	3.28×10^{0}	2.18×10^{0}	2.03×10^{0}
f15	Mean Error	5.42×10^{1} (6)	2.14×10^{1} (2)	8.34×10^{1} (8)	6.67×10^{1} (7)	**1.64×10^{1} (1)**	2.99×10^{1} (5)	2.81×10^{1} (4)	2.69×10^{1} (3)
	Std Dev	2.12×10^{1}	2.13×10^{0}	6.17×10^{1}	1.99×10^{1}	3.01×10^{0}	5.61×10^{0}	9.68×10^{0}	7.37×10^{0}
f16	Mean Error	5.41×10^{2} (6)	3.39×10^{2} (4)	8.15×10^{2} (7)	3.28×10^{2} (3)	3.53×10^{2} (5)	3.53×10^{2} (5)	**2.25×10^{2} (1)**	2.61×10^{2} (2)
	Std Dev	2.18×10^{2}	1.43×10^{2}	2.38×10^{2}	1.16×10^{2}	1.36×10^{2}	2.04×10^{2}	8.71×10^{1}	1.29×10^{2}
f17	Mean Error	4.44×10^{2} (7)	2.73×10^{2} (5)	4.55×10^{2} (8)	2.11×10^{2} (3)	2.09×10^{2} (2)	2.93×10^{2} (6)	**1.58×10^{2} (1)**	2.56×10^{2} (4)
	Std Dev	1.74×10^{2}	8.98×10^{1}	1.96×10^{2}	5.06×10^{1}	5.79×10^{1}	1.13×10^{2}	3.30×10^{1}	2.10×10^{2}
f18	Mean Error	1.19×10^{2} (8)	**2.35×10^{1} (1)**	1.04×10^{2} (7)	9.43×10^{1} (6)	3.19×10^{1} (4)	3.14×10^{1} (3)	7.35×10^{1} (5)	3.05×10^{1} (2)
	Std Dev	7.45×10^{1}	1.51×10^{0}	6.67×10^{1}	2.55×10^{1}	5.73×10^{0}	5.90×10^{0}	2.11×10^{1}	1.79×10^{1}
f19	Mean Error	2.85×10^{1} (7)	**1.10×10^{1} (1)**	1.15×10^{2} (8)	2.80×10^{1} (6)	1.45×10^{1} (2)	2.13×10^{1} (5)	1.90×10^{1} (4)	1.63×10^{1} (3)
	Std Dev	5.38×10^{0}	2.17×10^{0}	4.09×10^{1}	5.76×10^{0}	3.17×10^{0}	3.89×10^{0}	2.58×10^{0}	2.71×10^{0}
f20	Mean Error	3.14×10^{2} (7)	**1.39×10^{2} (1)**	3.22×10^{2} (8)	2.00×10^{2} (6)	1.68×10^{2} (2)	1.76×10^{2} (3)	1.80×10^{2} (4)	1.88×10^{2} (5)
	Std Dev	1.36×10^{2}	7.23×10^{1}	1.44×10^{2}	3.91×10^{1}	2.44×10^{1}	1.10×10^{2}	2.66×10^{1}	2.05×10^{1}
f21	Mean Error	2.52×10^{2} (7)	2.16×10^{2} (6)	2.73×10^{2} (8)	1.00×10^{2} (4)	**3.05×10^{1} (1)**	2.14×10^{2} (5)	3.73×10^{1} (2)	3.94×10^{1} (3)
	Std Dev	1.19×10^{1}	4.78×10^{0}	2.64×10^{1}	9.20×10^{-14}	2.95×10^{1}	9.77×10^{0}	3.51×10^{1}	1.66×10^{1}
f22	Mean Error	3.40×10^{2} (5)	1.02×10^{3} (6)	7.84×10^{3} (8)	1.00×10^{2} (4)	1.44×10^{1} (3)	1.67×10^{3} (7)	1.25×10^{1} (2)	**1.10×10^{1} (1)**
	Std Dev	9.71×10^{2}	1.78×10^{3}	3.34×10^{3}	9.20×10^{-14}	2.74×10^{0}	2.08×10^{3}	2.19×10^{0}	1.00×10^{0}
f23	Mean Error	4.71×10^{2} (5)	4.31×10^{2} (3)	4.89×10^{2} (6)	7.19×10^{2} (7)	7.20×10^{2} (8)	4.40×10^{2} (4)	**2.00×10^{2} (1)**	**2.00×10^{2} (1)**
	Std Dev	1.35×10^{1}	5.90×10^{0}	1.54×10^{1}	2.18×10^{1}	1.91×10^{1}	6.19×10^{0}	3.83×10^{-13}	1.79×10^{-11}
f24	Mean Error	5.35×10^{2} (6)	5.07×10^{2} (4)	5.68×10^{2} (8)	5.47×10^{2} (7)	3.00×10^{2} (3)	5.11×10^{2} (5)	2.33×10^{2} (2)	**2.18×10^{2} (1)**
	Std Dev	1.18×10^{1}	3.78×10^{0}	1.75×10^{1}	3.73×10^{2}	3.78×10^{-13}	5.74×10^{0}	2.40×10^{1}	1.59×10^{1}
f25	Mean Error	5.29×10^{2} (8)	4.81×10^{2} (3)	5.11×10^{2} (7)	5.02×10^{2} (6)	4.86×10^{2} (5)	4.81×10^{2} (3)	**2.00×10^{2} (1)**	**2.00×10^{2} (1)**
	Std Dev	2.14×10^{1}	2.80×10^{0}	3.19×10^{1}	2.83×10^{1}	2.72×10^{1}	2.75×10^{0}	0.00×10^{0}	0.00×10^{0}
f26	Mean Error	1.48×10^{3} (6)	1.08×10^{3} (4)	1.82×10^{3} (7)	2.97×10^{3} (8)	2.51×10^{2} (2)	1.15×10^{3} (5)	4.91×10^{2} (3)	**2.00×10^{2} (1)**
	Std Dev	1.23×10^{2}	5.29×10^{1}	2.16×10^{2}	7.07×10^{2}	5.05×10^{1}	5.79×10^{1}	2.58×10^{2}	0.00×10^{0}
f27	Mean Error	5.16×10^{2} (4)	5.08×10^{2} (3)	5.17×10^{2} (5)	8.77×10^{2} (7)	8.80×10^{2} (8)	5.31×10^{2} (6)	**2.13×10^{2} (1)**	2.64×10^{2} (2)
	Std Dev	8.35×10^{0}	8.70×10^{0}	1.12×10^{1}	3.60×10^{1}	4.12×10^{1}	1.26×10^{1}	9.17×10^{1}	1.09×10^{2}
f28	Mean Error	4.91×10^{2} (7)	4.59×10^{2} (4)	4.75×10^{2} (6)	5.30×10^{2} (8)	4.46×10^{2} (3)	4.60×10^{2} (5)	4.33×10^{2} (2)	**2.85×10^{2} (1)**
	Std Dev	2.33×10^{1}	1.82×10^{-13}	2.30×10^{1}	4.25×10^{1}	3.30×10^{1}	6.84×10^{0}	2.55×10^{1}	2.04×10^{1}

Table 6. *Cont.*

Func	Statistic	NPADE	IDE-DAE	FADE	MPEDE	ADE-DMRM	MMDE	DPMADE	DPR-MGDE
f29	Mean Error	3.44×10^{2} (3)	3.81×10^{2} (4)	4.04×10^{2} (6)	5.67×10^{2} (7)	5.85×10^{2} (8)	3.81×10^{2} (4)	2.62×10^{2} (2)	**2.00×10^{2} (1)**
	Std Dev	1.98×10^{1}	1.89×10^{1}	7.83×10^{1}	3.41×10^{1}	3.00×10^{1}	4.75×10^{1}	1.90×10^{2}	0.00×10^{0}
f30	Mean Error	6.19×10^{5} (7)	5.94×10^{5} (6)	5.93×10^{5} (5)	5.70×10^{3} (4)	5.56×10^{3} (3)	6.57×10^{5} (8)	4.73×10^{3} (2)	**2.00×10^{3} (1)**
	Std Dev	3.98×10^{4}	2.82×10^{4}	2.29×10^{4}	3.03×10^{2}	2.12×10^{2}	8.39×10^{4}	3.83×10^{3}	2.67×10^{2}
		28/2/0	20/10/0	30/0/0	24/6/0	20/10/0	2010/0	18/10/2	
	Rank	7.60	3.41	7.21	5.17	4.37	4.07	2.80	2.66

Bold data indicates the best results experimentally obtained on a certain test function.

Based on the data presented in Table 7, it is evident that the DPR-MGDE algorithm achieves the best average ranking across all benchmark functions, highlighting its overall superior performance in the optimisation tasks under consideration. In the category of single-objective functions (F1–F10), DPR-MGDE generally does not outperform the other comparison algorithms, with the exception of F10, where it demonstrates a notable improvement. This suggests that, while DPR-MGDE may face challenges in certain single-objective optimisation scenarios, it can still achieve competitive results in specific instances. In the hybrid functions (F11–F20), DPR-MGDE shows a marked improvement, achieving the best performance on F16 and F17. Additionally, it remains competitive for the other benchmark functions within this category, outperforming most of the comparison algorithms. This indicates DPR-MGDE's ability to effectively handle the complexity and multimodality inherent in hybrid functions, making it a robust choice for such optimisation problems. In the combined functions (F21–F30), DPR-MGDE consistently outperforms all other comparison algorithms, with the notable exceptions of F21 and F26. In these two instances, DPR-MGDE ranks second, trailing only behind ADE-DMRM and MMDE, respectively. This performance further underscores DPR-MGDE's strength in tackling complex, combined optimisation problems, where it can effectively navigate the intricacies of multiple interacting objectives. Overall, the results from Table 7 provide a comprehensive evaluation of DPR-MGDE's performance across a diverse set of optimisation functions. While it may not dominate in every single-objective function, its strong performance in hybrid and combined functions, particularly in achieving the best average ranking across all benchmark functions, confirms its robustness and adaptability. This makes DPR-MGDE a valuable algorithm for addressing a wide range of optimisation challenges, especially those involving high-complexity and multiple objectives.

Table 7. Experimental results of DPR-MGDE and 10 famous or up-to-date DE variants on CEC2017 test suite when D = 100.

Func	Statistic	NPADE	IDE-EDA	FADE	MPEDE	ADE-DMRM	MMDE	DPMADE	DPR-MGDE
f1	Mean Error	8.12×10^{-3} (7)	7.04×10^{-11} (6)	6.23×10^{3} (8)	3.13×10^{-12} (5)	1.00×10^{-13} (2)	1.52×10^{-13} (3)	**8.08×10^{-15} (1)**	2.80×10^{-12} (4)
	Std Dev	3.42×10^{-3}	2.10×10^{-10}	4.75×10^{3}	6.71×10^{-12}	3.52×10^{-14}	1.15×10^{-13}	7.11×10^{-15}	2.43×10^{-12}
f2	Mean Error	-	-	-	-	-	-	-	-
	Std Dev	-	-	-	-	-	-	-	-
f3	Mean Error	8.96×10^{4} (8)	3.72×10^{-6} (5)	2.21×10^{2} (7)	4.12×10^{-5} (4)	2.23×10^{-8} (3)	2.85×10^{-9} (2)	**2.23×10^{-15} (1)**	4.54×10^{-6} (6)
	Std Dev	9.34×10^{4}	3.88×10^{-6}	1.25×10^{2}	9.46×10^{-5}	4.83×10^{-8}	2.38×10^{-9}	1.11×10^{-14}	4.28×10^{-6}

Table 7. *Cont.*

Func	Statistic	NPADE	IDE-EDA	FADE	MPEDE	ADE-DMRM	MMDE	DPMADE	DPR-MGDE
f4	Mean Error	1.12×10^{2} (2)	1.95×10^{2} (7)	1.43×10^{2} (4)	**4.39×10^{1} (1)**	1.55×10^{2} (5)	1.65×10^{2} (6)	2.06×10^{2} (8)	1.21×10^{2} (3)
	Std Dev	4.30×10^{1}	2.57×10^{1}	5.55×10^{1}	4.46×10^{1}	3.19×10^{1}	3.06×10^{1}	9.34×10^{0}	3.80×10^{1}
f5	Mean Error	1.17×10^{2} (6)	5.40×10^{1} (5)	1.94×10^{2} (8)	1.17×10^{2} (6)	4.62×10^{1} (4)	4.30×10^{1} (3)	**1.15×10^{1} (1)**	4.26×10^{1} (2)
	Std Dev	2.24×10^{1}	9.84×10^{0}	2.00×10^{1}	1.73×10^{1}	5.40×10^{0}	5.41×10^{0}	3.34×10^{0}	5.18×10^{0}
f6	Mean Error	2.76×10^{-5} (2)	1.97×10^{-4} (4)	3.03×10^{-3} (5)	1.68×10^{0} (8)	6.62×10^{-3} (6)	1.19×10^{-4} (3)	**1.14×10^{-13} (1)**	9.03×10^{-3} (7)
	Std Dev	2.33×10^{-5}	6.91×10^{-4}	9.00×10^{-3}	6.18×10^{-1}	5.32×10^{-3}	3.16×10^{-4}	0.00×10^{0}	6.88×10^{-3}
f7	Mean Error	2.00×10^{2} (6)	1.65×10^{2} (5)	3.09×10^{2} (8)	2.65×10^{2} (7)	1.50×10^{2} (4)	1.45×10^{2} (2)	**1.12×10^{2} (1)**	1.46×10^{2} (3)
	Std Dev	2.14×10^{1}	8.95×10^{0}	2.97×10^{1}	2.66×10^{1}	5.65×10^{0}	5.59×10^{0}	1.65×10^{0}	5.61×10^{0}
f8	Mean Error	1.24×10^{2} (6)	4.75×10^{1} (4)	1.83×10^{2} (8)	1.31×10^{2} (7)	4.89×10^{1} (5)	4.44×10^{1} (2)	**1.12×10^{1} (1)**	4.62×10^{1} (3)
	Std Dev	2.48×10^{1}	1.20×10^{1}	2.58×10^{1}	1.75×10^{1}	6.04×10^{0}	4.97×10^{0}	2.93×10^{0}	4.85×10^{0}
f9	Mean Error	2.21×10^{0} (6)	5.12×10^{-2} (3)	1.31×10^{2} (7)	5.02×10^{2} (8)	3.67×10^{-1} (4)	1.05×10^{-2} (2)	**0.00×10^{0} (1)**	5.46×10^{-1} (5)
	Std Dev	1.66×10^{0}	1.13×10^{-1}	8.80×10^{1}	2.19×10^{2}	3.86×10^{-1}	3.42×10^{-2}	0.00×10^{0}	5.78×10^{-1}
f10	Mean Error	1.24×10^{4} (8)	1.16×10^{4} (5)	1.19×10^{4} (6)	1.22×10^{4} (7)	1.06×10^{4} (4)	9.91×10^{3} (3)	9.89×10^{3} (2)	**9.47×10^{3} (1)**
	Std Dev	1.44×10^{3}	6.48×10^{2}	1.32×10^{3}	1.13×10^{3}	5.25×10^{2}	6.28×10^{2}	9.80×10^{2}	5.05×10^{2}
f11	Mean Error	2.45×10^{3} (8)	9.06×10^{1} (2)	2.00×10^{2} (4)	5.48×10^{2} (7)	3.36×10^{2} (6)	1.45×10^{2} (3)	**5.05×10^{1} (1)**	3.18×10^{2} (5)
	Std Dev	1.68×10^{3}	2.81×10^{1}	7.23×10^{1}	1.20×10^{2}	4.91×10^{1}	2.04×10^{1}	2.32×10^{1}	5.69×10^{1}
f12	Mean Error	3.37×10^{4} (7)	1.48×10^{4} (6)	1.49×10^{5} (8)	6.18×10^{3} (5)	5.40×10^{3} (4)	**4.73×10^{3} (1)**	4.91×10^{3} (2)	5.38×10^{3} (3)
	Std Dev	1.42×10^{4}	7.33×10^{3}	4.54×10^{4}	1.63×10^{3}	9.33×10^{2}	9.14×10^{2}	8.30×10^{2}	9.68×10^{2}
f13	Mean Error	3.34×10^{2} (5)	1.36×10^{2} (2)	2.94×10^{3} (8)	1.40×10^{3} (7)	3.52×10^{2} (6)	1.92×10^{2} (4)	**1.35×10^{2} (1)**	1.82×10^{2} (3)
	Std Dev	8.85×10^{1}	3.70×10^{1}	2.38×10^{3}	8.12×10^{2}	9.01×10^{1}	5.29×10^{1}	3.59×10^{1}	5.78×10^{1}
f14	Mean Error	2.57×10^{2} (6)	8.01×10^{1} (4)	1.93×10^{2} (5)	5.31×10^{2} (8)	5.03×10^{2} (7)	**4.03×10^{1} (1)**	7.70×10^{1} (3)	5.14×10^{1} (2)
	Std Dev	4.64×10^{1}	6.73×10^{0}	4.79×10^{1}	9.43×10^{1}	4.46×10^{1}	5.35×10^{0}	1.37×10^{1}	5.58×10^{1}
f15	Mean Error	3.40×10^{2} (6)	1.28×10^{2} (2)	1.75×10^{3} (8)	3.13×10^{2} (5)	3.57×10^{2} (7)	**7.20×10^{1} (1)**	2.13×10^{2} (4)	1.73×10^{2} (3)
	Std Dev	7.06×10^{1}	3.28×10^{1}	2.02×10^{3}	1.32×10^{2}	6.08×10^{1}	1.51×10^{1}	3.06×10^{1}	6.26×10^{1}
f16	Mean Error	2.51×10^{3} (8)	1.60×10^{3} (2)	2.29×10^{3} (5)	2.41×10^{3} (6)	1.94×10^{3} (4)	1.88×10^{3} (3)	2.42×10^{3} (7)	**1.30×10^{3} (1)**
	Std Dev	3.97×10^{2}	3.47×10^{2}	5.86×10^{2}	6.76×10^{2}	2.71×10^{2}	2.88×10^{2}	4.47×10^{2}	2.13×10^{2}
f17	Mean Error	1.77×10^{3} (8)	1.24×10^{3} (6)	1.54×10^{3} (7)	1.04×10^{3} (4)	6.36×10^{2} (2)	7.26×10^{2} (3)	1.14×10^{3} (5)	**5.53×10^{2} (1)**
	Std Dev	3.90×10^{2}	2.31×10^{2}	2.70×10^{2}	3.64×10^{2}	1.30×10^{2}	1.68×10^{2}	3.32×10^{2}	1.05×10^{2}
f18	Mean Error	2.67×10^{2} (6)	1.36×10^{2} (2)	8.09×10^{3} (8)	2.60×10^{3} (5)	2.76×10^{2} (7)	2.31×10^{2} (3)	**1.24×10^{2} (1)**	2.48×10^{2} (4)
	Std Dev	6.50×10^{1}	3.09×10^{1}	4.84×10^{3}	2.46×10^{3}	5.87×10^{1}	3.85×10^{1}	3.56×10^{1}	7.49×10^{1}
f19	Mean Error	4.12×10^{2} (5)	8.35×10^{1} (7)	8.46×10^{2} (8)	4.43×10^{2} (6)	4.01×10^{2} (4)	**7.40×10^{1} (1)**	**7.40×10^{1} (1)**	8.09×10^{1} (3)
	Std Dev	7.94×10^{1}	1.42×10^{1}	5.79×10^{2}	1.23×10^{2}	5.71×10^{1}	1.14×10^{1}	1.06×10^{1}	5.53×10^{1}
f20	Mean Error	2.01×10^{3} (8)	1.55×10^{3} (5)	1.63×10^{3} (7)	1.58×10^{3} (6)	1.07×10^{3} (3)	**9.15×10^{2} (1)**	1.37×10^{3} (4)	9.79×10^{2} (2)
	Std Dev	4.36×10^{2}	2.54×10^{2}	1.84×10^{2}	3.51×10^{2}	1.40×10^{2}	1.63×10^{2}	3.90×10^{2}	2.02×10^{2}
f21	Mean Error	3.38×10^{2} (7)	2.57×10^{2} (6)	4.10×10^{2} (8)	1.50×10^{2} (2)	**1.40×10^{2} (1)**	1.50×10^{2} (2)	2.42×10^{2} (5)	1.50×10^{2} (2)
	Std Dev	2.16×10^{1}	1.30×10^{1}	2.88×10^{1}	1.04×10^{-12}	5.06×10^{1}	2.17×10^{-13}	9.14×10^{0}	6.89×10^{1}

Table 7. *Cont.*

Func	Statistic	NPADE	IDE-EDA	FADE	MPEDE	ADE-DMRM	MMDE	DPMADE	DPR-MGDE
f22	Mean Error	9.12×10^{3} (5)	1.24×10^{4} (8)	1.20×10^{4} (7)	1.58×10^{2} (4)	6.37×10^{1} (2)	1.41×10^{2} (3)	9.96×10^{3} (6)	**4.31×10^{1} (1)**
	Std Dev	5.95×10^{3}	6.64×10^{2}	1.08×10^{3}	7.30×10^{-13}	2.63×10^{1}	1.75×10^{-13}	8.84×10^{2}	3.84×10^{0}
f23	Mean Error	6.58×10^{2} (4)	5.65×10^{2} (2)	6.95×10^{2} (5)	1.29×10^{3} (8)	1.15×10^{3} (7)	1.13×10^{3} (6)	5.81×10^{2} (3)	**2.00×10^{2} (1)**
	Std Dev	2.47×10^{1}	1.11×10^{1}	3.66×10^{1}	5.79×10^{1}	3.26×10^{1}	3.88×10^{1}	7.09×10^{0}	8.67×10^{-14}
f24	Mean Error	9.95×10^{2} (7)	8.97×10^{2} (5)	1.07×10^{3} (8)	5.17×10^{2} (4)	2.08×10^{2} (2)	3.06×10^{2} (3)	9.14×10^{2} (6)	**2.56×10^{2} (1)**
	Std Dev	2.74×10^{1}	6.87×10^{0}	3.94×10^{1}	5.61×10^{-13}	2.72×10^{1}	2.16×10^{-13}	2.11×10^{1}	1.92×10^{2}
f25	Mean Error	7.73×10^{2} (6)	7.19×10^{2} (4)	7.89×10^{2} (7)	8.16×10^{2} (8)	7.70×10^{2} (5)	6.99×10^{2} (3)	6.92×10^{2} (2)	**3.59×10^{2} (1)**
	Std Dev	3.46×10^{1}	4.23×10^{1}	4.83×10^{1}	5.68×10^{1}	5.81×10^{1}	4.76×10^{1}	4.60×10^{1}	3.11×10^{1}
f26	Mean Error	4.10×10^{3} (6)	3.09×10^{3} (4)	5.12×10^{3} (7)	2.65×10^{3} (3)	7.97×10^{3} (8)	**2.88×10^{2} (1)**	3.13×10^{3} (5)	2.00×10^{3} (2)
	Std Dev	2.78×10^{2}	7.88×10^{1}	4.39×10^{2}	4.46×10^{3}	1.75×10^{2}	3.25×10^{1}	5.92×10^{1}	2.57×10^{1}
f27	Mean Error	5.98×10^{2} (3)	5.78×10^{2} (2)	6.15×10^{2} (5)	1.51×10^{3} (6)	1.66×10^{3} (8)	1.62×10^{3} (7)	6.01×10^{2} (4)	**2.54×10^{2} (1)**
	Std Dev	1.83×10^{1}	1.97×10^{1}	2.13×10^{1}	8.56×10^{2}	1.15×10^{2}	1.84×10^{2}	1.66×10^{1}	2.93×10^{2}
f28	Mean Error	5.65×10^{2} (8)	5.27×10^{2} (6)	5.55×10^{2} (7)	4.95×10^{2} (2)	5.03×10^{2} (3)	5.06×10^{2} (4)	5.17×10^{2} (5)	**2.00×10^{2} (1)**
	Std Dev	2.21×10^{1}	3.07×10^{1}	2.09×10^{1}	9.73×10^{0}	1.50×10^{1}	1.37×10^{1}	1.91×10^{1}	0.00×10^{0}
f29	Mean Error	1.34×10^{3} (6)	1.15×10^{3} (4)	1.90×10^{3} (8)	1.21×10^{3} (5)	9.39×10^{2} (2)	9.86×10^{2} (3)	1.50×10^{3} (7)	**2.00×10^{2} (1)**
	Std Dev	3.03×10^{2}	1.83×10^{2}	3.49×10^{2}	3.71×10^{2}	8.41×10^{1}	1.06×10^{2}	3.35×10^{2}	0.00×10^{0}
f30	Mean Error	2.33×10^{3} (4)	2.18×10^{3} (3)	3.57×10^{3} (7)	6.13×10^{2} (2)	3.59×10^{3} (8)	3.46×10^{3} (6)	2.42×10^{3} (5)	**2.00×10^{2} (1)**
	Std Dev	1.28×10^{2}	7.58×10^{1}	1.16×10^{3}	7.95×10^{1}	1.32×10^{2}	1.33×10^{2}	1.71×10^{2}	0.00×10^{0}
	b/w/s	28/2/0	23/7/0	28/2/0	27/1/2	26/4/0	16/13/1	18/12/0	-
	Rank	6.00	4.69	6.82	5.37	4.59	2.93	3.24	2.52

Bold data indicates the best results experimentally obtained on a certain test function.

Moreover, by applying the multi-problem Wilcoxon signed rank test, we further validate the superiority of the DPR-MGDE algorithm over other optimisation algorithms. Table 8 details the statistical comparison results between DPR-MGDE and other algorithms under different dimension settings. When the problem dimension D = 10, DPR-MGDE exhibits a significant performance difference with all the compared algorithms at the 0.05 significance level. This indicates that the performance improvement of the DPR-MGDE algorithm is statistically significant in lower-dimensional optimisation problems. When the dimensionality is raised to D = 30, DPR-MGDE shows the same significant difference with all other rivals except for IDE-EDA. When dimension D = 50, DPR-MGDE maintains its competitiveness and performs better in most cases, except when compared to ADE-DMRM and DPMADE. In the case of higher dimension (D = 100), DPR-MGDE has a p-value of less than 0.05 when compared to all other algorithms except MMDE, which further confirms the significant improvement and efficacy of DPR-MGDE in dealing with high-dimensional optimisation problems.

Table 8. Comparison results of DPR-MGDE with 10 famous or up-to-date DE variants based on the multiproblem Wilcoxon signed-rank test on CEC2017 test suite.

DPR-MGDE VS	**D = 10**		**D = 30**		**D = 50**		**D = 100**	
	ρ-Value	**α = 0.05**	**ρ-Value**	**α = 0.05**	**ρ-Value**	**α = 0.05**	**ρ-Value**	**α = 0.05**
NPADE	0.0490	YES	0.0011	YES	3.5104×10^{-6}	YES	4.8009×10^{-6}	YES
IDE-EDA	0.0299	YES	0.1023	NO	0.0121	YES	6.0967×10^{-4}	YES
FADE	0.0039	YES	2.5614×10^{-6}	YES	2.5631×10^{-6}	YES	3.9017×10^{-6}	YES
MPEDE	1.1805×10^{-5}	YES	5.3218×10^{-6}	YES	3.9067×10^{-4}	YES	6.5213×10^{-6}	YES
ADE-DMRM	1.2320×10^{-5}	YES	0.0013	YES	0.0856	NO	6.0436×10^{-5}	YES
MMDE	0.0027	YES	0.0186	YES	0.0118	YES	0.2319	NO
DPMADE	0.4286×10^{-5}	YES	0.0071	YES	0.2297	NO	0.0118	YES

Taken together, these results show that the DPR-MGDE algorithm proposed in this paper achieves significant performance gains over a wide range of dimensions, compared to a variety of existing optimisation algorithms. This finding demonstrates the effectiveness and reliability of DPR-MGDE in solving complex optimisation problems, especially its robust performance in the face of high-dimensionality challenges. This may be due to the fact that the multiple sub-populations divided by the dual performance evaluation metrics of DPR-MGDE are able to distinguish the potentials of individuals, use different variation strategies for them and adaptively update the variation factors and crossover factors according to the changes of the individuals, so as to obtain better optimal solutions. And due to the inclusion of a restarting strategy, which helps the algorithm to jump out of the local optimum and maintain the diversity of the populations, DPR-MGDE can achieve significant performance improvements in solving complex optimisation problems, especially when facing high-dimensional challenges. On this basis, DPR-MGDE can show better performance when dealing with optimisation problems with high latitude complexity.

4.4. Real Application

In this part of the study, we aim to demonstrate the broad utility of the DPR-MGDE algorithm by applying it to problems in five different engineering domains presented in the literature [40]. These problems include the multi-product intermittent plant problem (P01) in the mechanical design domain, the chemical engineering problem (P02), the process design and synthesis problem (P03), the power electronics problem (P04) and the power system problem (P05). Specifically, P01 deals with the optimisation of a multi-product intermittent plant, P02 deals with reactor network design, P03 is the optimisation of a two-reactor problem, P04 is the synchronous optimal pulse width modulation of a three-level inverter and P05 is the problem of optimal allocation of reactive power support for a single-phase distribution network based on minimising active and reactive losses. A detailed description of these problems can be found in reference [40]. In order to validate the performance of the DPR-MGDE algorithm, we have selected the comparison algorithms discussed in Section 4.3 as competing algorithms and performed a comprehensive comparison. According to the results shown in Table 9, DPR-MGDE achieves the best performance on three real engineering problems, P01, P04 and P05. On problem P02, DPR-MGDE outperforms DPMADE, IDE-EDA, FADE and NPADE, while on problem P03, it outperforms NPADE, FADE and IDE-EDA. In addition, the data show that DPR-MGDE outperforms at least half of the compared algorithms on all five problems, and on some of them it received the highest rankings. These results fully demonstrate the effectiveness and superiority of DPR-MGDE in solving these real-world engineering problems.

Table 9. Numerical and comparison results of DPMADE and up-to-date DE variants on real-world problems.

Problem	Statistic	NPADE	IDE-EDA	FADE	MPEDE	ADE-DMRM	MMDE	DPMADE	DPR-MGDE
P01	Mean Error	2.30×10^{4} (7)	1.76×10^{4} (6)	2.57×10^{4} (8)	7.03×10^{3} (3)	6.92×10^{3} (2)	9.24×10^{3} (5)	7.19×10^{3} (4)	$\mathbf{6.79 \times 10^{3}}$ **(1)**
	Std Dev	3.17×10^{2}	1.28×10^{2}	1.91×10^{2}	4.64×10^{2}	2.83×10^{1}	5.57×10^{2}	3.21×10^{2}	4.45×10^{2}
P02	Mean Error	6.67×10^{0} (8)	5.19×10^{0} (6)	5.63×10^{0} (7)	3.51×10^{-1} (3)	$\mathbf{-5.47 \times 10^{-4}}$ **(1)**	6.43×10^{-3} (2)	4.98×10^{0} (5)	4.02×10^{0} (4)
	Std Dev	2.96×10^{0}	3.06×10^{0}	4.73×10^{0}	7.07×10^{-1}	2.72×10^{0}	2.94×10^{-1}	3.08×10^{0}	2.05×10^{0}
P03	Mean Error	3.35×10^{1} (6)	4.00×10^{1} (8)	3.72×10^{1} (7)	2.30×10^{1} (4)	2.08×10^{1} (3)	1.97×10^{1} (2)	$\mathbf{1.58 \times 10^{1}}$ **(1)**	2.39×10^{1} (5)
	Std Dev	2.18×10^{1}	2.55×10^{1}	2.03×10^{0}	1.94×10^{0}	1.56×10^{0}	1.02×10^{1}	1.75×10^{0}	1.98×10^{1}
P04	Mean Error	4.42×10^{0} (7)	3.75×10^{0} (6)	2.46×10^{0} (4)	6.83×10^{0} (8)	2.95×10^{0} (5)	3.88×10^{-1} (2)	1.16×10^{0} (3)	$\mathbf{1.26 \times 10^{-1}}$ **(1)**
	Std Dev	7.84×10^{-1}	2.11×10^{0}	1.90×10^{0}	5.05×10^{0}	2.07×10^{-1}	1.59×10^{0}	2.04×10^{-1}	3.47×10^{-1}
P05	Mean Error	1.93×10^{-1} (6)	4.06×10^{-1} (7)	6.08×10^{-1} (8)	7.88×10^{-2} (4)	3.49×10^{-2} (2)	3.91×10^{-2} (3)	9.04×10^{-2} (5)	$\mathbf{4.01 \times 10^{-3}}$ **(1)**
	Std Dev	2.33×10^{-1}	3.17×10^{-1}	4.42×10^{-1}	5.61×10^{-2}	3.02×10^{-2}	1.08×10^{-1}	1.39×10^{-1}	3.86×10^{-2}
b/w/s		5/0/0	5/0/0	5/0/0	3/2/0	3/2/0	3/2/0	4/1/0	-
Rank		6.6	6.6	6.6	4.4	2.6	2.8	3.6	2.4

Bold data indicates the best results experimentally obtained on engineering issues.

5. Conclusions

In this paper, a novel dual-performance multiple sub-population adaptive restart differential evolutionary algorithm (DPR-MGDE) is proposed to solve the local voptimum trap and balance global exploration and local exploitation in high-dimensional complex multi-peak optimisation problems. The DPR-MGDE algorithm divides the population into three different sub-populations (PM, MM, and UM). The three subpopulations play different roles due to the different variation strategies used by them: the PM subpopulation focuses on the exploration of new scenarios, the MM subpopulation focuses on population development and the UM subpopulation is responsible for maintaining the diversity of the population by adding new individuals when searching is stagnant, thus balancing exploration and development. In addition, the adaptive cross-covariance strategy and collision-based Gaussian wandering restart strategy in the algorithm further improve the flexibility of the algorithm and the ability to jump out of local optima. Experimental validation on the CEC2017 benchmark functions shows that the DPR-MGDE algorithm outperforms other state-of-the-art variants of differential evolution algorithms in terms of solution accuracy. This is most likely due to the fact that we are better able to jump out of the local optimum in the multi-peak problem, while the adaptive parameter settings enable the algorithm to better adapt to different dimensions of the test function and have better flexibility, in addition to the fact that the multi-subpopulation approach allows the algorithm to maintain a good search capability on different dimensions of the test function. Therefore, the algorithm has significant advantages in terms of global exploration, optimisation accuracy and maintaining population diversity, especially when dealing with high-latitude complex optimisation problems. Compared to several existing variants of differential evolutionary algorithms, DPR-MGDE exhibits superior performance on test functions of different dimensions. Statistical analysis using Wilcoxon's rank sum test further confirms that the improved performance of the DPR-MGDE algorithm is statistically significant when solving complex high-dimensional optimisation problems. In addition, experiments on real-world problems show that DPR-MGDE is effective and superior for solving problems in five different engineering domains.

In conclusion, the DPR-MGDE algorithm effectively overcomes the limitations of traditional differential evolution algorithms, can effectively jump out of the local optimum and at the same time has a good search ability in high-dimensional complex problems, in addition to balancing the exploration and exploitation and maintaining the diversity of the population. It demonstrates its strengths in solving multiple aspects of complex optimisation problems and practical problems. Future work focuses on further optimising the algorithm parameters to make it more adaptable and able to be applied to more scenarios.

Author Contributions: Conceptualisation, Y.S. and Y.X.; methodology, Y.S.; validation, Y.X.; formal analysis, Y.S. and Q.C.; data curation, Y.X.; writing—original draft preparation, Y.X.; writing—review and editing, Y.S. and Q.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data are available from the authors upon reasonable request.

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

References

1. Pant, M.; Zaheer, H.; Garcia-Hernandez, L.; Abraham, A.; Bilal. Differential Evolution: A review of more than two decades of research. *Eng. Appl. Artif. Intell.* **2020**, *90*, 103479.
2. Mohamed, A.W. A novel differential evolution algorithm for solving constrained engineering optimization problems. *J. Intell. Manuf.* **2018**, *29*, 659–692. [CrossRef]
3. Wang, D.; Sun, X.; Kang, H.; Shen, Y.; Chen, Q. Heterogeneous differential evolution algorithm for parameter estimation of solar photovoltaic models. *Energy Rep.* **2022**, *8*, 4724–4746. [CrossRef]
4. Sharma, M.; Komninos, A.; López-Ibáñez, M.; Kazakov, D. Deep reinforcement learning based parameter control in differential evolution. In Proceedings of the Genetic and Evolutionary Computation Conference, Prague, Czech Republic, 13–17 July 2019; pp. 709–717.
5. Sun, X.; Zhang, T.; Xu, J.; Zhang, H.; Kang, H.; Shen, Y.; Chen, Q. Energy efficiency-driven mobile base station deployment strategy for shopping malls using modified improved differential evolution algorithm. *Appl. Intell.* **2023**, *53*, 1233–1253. [CrossRef]
6. Du, J.X.; Huang, D.S.; Wang, X.F.; Gu, X. Shape recognition based on neural networks trained by differential evolution algorithm. *Neurocomputing* **2007**, *70*, 896–903. [CrossRef]
7. Shen, Y.; Chen, Y.; Kang, H.; Sun, X.; Chen, Q. Energy-efficient indoor hybrid deployment strategy for 5G mobile small-cell base stations using JAFR Algorithm. *Pervasive Mob. Comput.* **2024**, *100*, 101918. [CrossRef]
8. Das, S.; Mullick, S.S.; Suganthan, P.N. Recent advances in differential evolution—An updated survey. *Swarm Evol. Comput.* **2016**, *27*, 1–30. [CrossRef]
9. Sá, Â.A.; Andrade, A.O.; Soares, A.B.; Nasuto, S.J. Exploration vs. exploitation in differential evolution. In Proceedings of the AISB 2008 Convention Communication, Interaction and Social Intelligence, Aberdeen, UK, 1–4 April 2008; Volume 1, p. 57.
10. Ahmad, M.F.; Isa, N.A.M.; Lim, W.H.; Ang, K.M. Differential evolution: A recent review based on state-of-the-art works. *Alex. Eng. J.* **2022**, *61*, 3831–3872. [CrossRef]
11. Deng, W.; Shang, S.; Cai, X.; Zhao, H.;Song, Y.; Xu, J. An improved differential evolution algorithm and its application in optimization problem. *Soft Comput.* **2021**, *25*, 5277–5298. [CrossRef]
12. Al-Dabbagh, R.D.; Neri, F.; Idris, N.; Baba, M.S. Algorithmic design issues in adaptive differential evolution schemes: Review and taxonomy. *Swarm Evol. Comput.* **2018**, *43*, 284–311. [CrossRef]
13. Zhang, J.; Sanderson, A.C. JADE: Adaptive differential evolution with optional external archive. *IEEE Trans. Evol. Comput.* **2009**, *13*, 945–958. [CrossRef]
14. Leon, M.; Xiong, N. Adaptive differential evolution with a new joint parameter adaptation method. *Soft Comput.* **2020**, *24*, 12801–12819. [CrossRef]
15. Wang, H.B.; Ren, X.N.; Li, G.Q.; Tu, X.Y. APDDE: Self-adaptive parameter dynamics differential evolution algorithm. *Soft Comput.* **2018**, *22*, 1313–1333. [CrossRef]
16. Fan, Q.; Yan, X. Self-adaptive differential evolution algorithm with discrete mutation control parameters. *Expert Syst. Appl.* **2015**, *42*, 1551–1572. [CrossRef]
17. Shen, Y.; Li, Y.; Kang, H.; Zhang, Y.; Sun, X.; Chen, Q.; Peng, J.; Wang, H. Research on swarm size of multi-swarm particle swarm optimization algorithm. In Proceedings of the 2018 IEEE 4th International Conference on Computer and Communications (ICCC), Chengdu, China, 7–10 December 2018; pp. 2243–2247.
18. Wu, G.; Mallipeddi, R.; Suganthan, P.N.; Wang, R.; Chen, H. Differential evolution with multi-population based ensemble of mutation strategies. *Inf. Sci.* **2016**, *329*, 329–345. [CrossRef]

19. Zhu, L.; Ma, Y.; Bai, Y. A self-adaptive multi-population differential evolution algorithm. *Nat. Comput.* **2020**, *19*, 211–235. [CrossRef]
20. Song, Y.; Wu, D.; Deng, W.; Gao, X.Z.; Li, T.; Zhang, B.; Li, Y. MPPCEDE: Multi-population parallel co-evolutionary differential evolution for parameter optimization. *Energy Convers. Manag.* **2021**, *228*, 113661. [CrossRef]
21. Liu, Q.; Pang, T.; Chen, K.; Wang, Z.; Sheng, W. Adaptive multi-subpopulation based differential evolution for global optimization. In Proceedings of the 2022 IEEE Congress on Evolutionary Computation (CEC), Padua, Italy, 18–23 July 2022; pp. 1–7.
22. Sun, X.; Wang, D.; Kang, H.; Shen, Y.; Chen, Q. A two-stage differential evolution algorithm with mutation strategy combination. *Symmetry* **2021**, *13*, 2163. [CrossRef]
23. Lin, X.; Meng, Z. An adaptative differential evolution with enhanced diversity and restart mechanism. *Expert Syst. Appl.* **2024**, *249*, 123634. [CrossRef]
24. Zhang, Q.; Meng, Z. Adaptive differential evolution algorithm based on deeply-informed mutation strategy and restart mechanism. *Eng. Appl. Artif. Intell.* **2023**, *126*, 107001. [CrossRef]
25. Tian, M.; Gao, X.; Yan, X. An improved differential evolution with a novel restart mechanism. In Proceedings of the 2016 12th International Conference on Computational Intelligence and Security (CIS), Wuxi, China, 16–19 December 2016; pp. 28–32.
26. Tian, M.; Yan, X.; Gao, X. An enhanced adaptive differential evolution algorithm with dual performance evaluation metrics for numerical optimization. *Swarm Evol. Comput.* **2024**, *84*, 101454. [CrossRef]
27. Shylo, O.V.; Prokopyev, O.A.; Rajgopal, J. On algorithm portfolios and restart strategies. *Oper. Res. Lett.* **2011**, *39*, 49–52. [CrossRef]
28. Cao, Y.; Luan, J. A novel differential evolution algorithm with multi-population and elites regeneration. *PLoS ONE* **2024**, *19*, e0302207. [CrossRef] [PubMed]
29. Chen, D.; Zou, F. A large-scale multi-objective optimization based on multi-population and multi-strategy differential algorithm. *Control Decis.* **2024**, *39*, 429–439.
30. Zhabitskaya, E.; Zhabitsky, M. Asynchronous differential evolution with restart. In *Numerical Analysis and Its Applications, Proceedings of the 5th International Conference, NAA 2012, Lozenetz, Bulgaria, 15–20 June 2012, Revised Selected Papers 5*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 555–561.
31. Poláková, R.; Tvrdík, J.; Bujok, P. Controlled restart in differential evolution applied to CEC2014 benchmark functions. In Proceedings of the 2014 IEEE Congress on Evolutionary Computation (CEC), Beijing, China, 6–11 July 2014; pp. 2230–2236.
32. Kitamura, T.; Fukunaga, A. Duplicate individuals in differential evolution. In Proceedings of the 2022 IEEE Congress on Evolutionary Computation (CEC), Padua, Italy, 18–23 July 2022; pp. 1–8.
33. Salimi, H. Stochastic fractal search: A powerful metaheuristic algorithm. *Knowl.-Based Syst.* **2015**, *75*, 1–18. [CrossRef]
34. Wu, G.; Mallipeddi, R.; Suganthan, P.N. *Problem Definitions and Evaluation Criteria for the CEC 2017 Competition on Constrained Real-Parameter Optimization*; Technical Report; National University of Defense Technology: Changsha, China; Kyungpook National University: Daegu, Republic of Korea; Nanyang Technological University: Singapore, 2017.
35. Wilcoxon, F. Individual comparisons by ranking methods. In *Breakthroughs in Statistics: Methodology and Distribution*; Springer: New York, NY, USA, 1992; pp. 196–202.
36. Tian, M.; Gao, X. Differential evolution with neighborhood-based adaptive evolution mechanism for numerical optimization. *Inf. Sci.* **2019**, *478*, 422–448. [CrossRef]
37. Li, Y.; Han, T.; Tang, S.; Huang, C.; Zhou, H.; Wang, Y. An improved differential evolution by hybridizing with estimation-of-distribution algorithm. *Inf. Sci.* **2023**, *619*, 439–456. [CrossRef]
38. Xia, X.; Gui, L.; Zhang, Y.; Xu, X.; Yu, F.; Wu, H.; Wei, B.; He, G.; Li, Y.; Li, K. A fitness-based adaptive differential evolution algorithm. *Inf. Sci.* **2021**, *549*, 116–141. [CrossRef]
39. Wang, K.; Wang, Y.; Tao, S.; Cai, Z.; Lei, Z.; Gao, S. Spherical search algorithm with adaptive population control for global continuous optimization problems. *Appl. Soft Comput.* **2023**, *132*, 109845. [CrossRef]
40. Kumar, A.; Wu, G.; Ali, M.Z.; Luo, Q.; Mallipeddi, R.; Suganthan, P.N.; Das, S. A benchmark-suite of real-world constrained multi-objective optimization problems and some baseline results. *Swarm Evol. Comput.* **2021**, *67*, 100961. [CrossRef]