The Journal of Supercomputing (2024) 80:22245–22286
https://doi.org/10.1007/s11227-024-06298-0

Enhancing differential evolution algorithm with a fitness-distance-based selection strategy

Yawei Huang[1] · Xuezhong Qian[1] · Wei Song[1]

Accepted: 9 June 2024 / Published online: 22 June 2024

Abstract

Since the introduction of differential evolution (DE) algorithms, they have achieved remarkable success in the field of evolutionary algorithms and engineering applications. In single-objective DE algorithms, most researchers tend to focus on improving mutation operators and parameter control, while overlooking the study of selection operators. However, the study of selection operators still holds great potential in enhancing the performance of DE algorithms. This study proposes a fitness-distance-based selection (FDS) strategy and a new scaling factor control method. FDS is divided into two stages. The first stage is to determine whether an individual needs to accept discarded trial vectors. The second stage involves selectively accepting these discarded trial vectors, which is based on the information related to the discarded trial vectors and the corresponding target vector. A new setting for the scaling factor parameter is proposed, designed to more effectively assist FDS in enhancing algorithm performance. Based on these strategies, an improved variant of the DE algorithm, called fitness-distance-based DE (FDDE) algorithm, is further proposed by this study. To verify the performance of FDDE, we conducted an in-depth study comparing it with six other advanced DE variants and four famous evolutionary algorithms using the CEC 2017, CEC 2022, and CEC 2011 benchmark sets. The experimental results demonstrate that the FDS strategy and the new scaling factor can significantly improve the performance of DE algorithms, and FDDE is significantly better than other advanced algorithm.

Keywords Differential evolution · Selection strategy · Scaling factor · Numerical optimization

✉ Yawei Huang
6213113063@stu.jiangnan.edu.cn

[1] School of Artificial Intelligence and Computer Science, Jiangnan university, Lihu Avenuc, Wuxi 214122, Jiangsu, China

1 Introduction

Storn and Price [1] introduced a population-based optimization technique known as Differential Evolution (DE). It can widely solve various optimization problems, including continuous optimization [2], discrete optimization [3], constrained optimization [4], and unconstrained optimization problems [5]. DE has also achieved success in practical applications [6–9]. However, like other evolutionary algorithms, DE is prone to get stuck in local optima [10, 11], and its performance still needs improvement.

Mutation, crossover [12, 13], and selection operators are prominent features of the DE algorithm. These three operations greatly influence DE's efficiency. Among them, the mutation operation has garnered significant attention from researchers and undergone extensive investigation. Typically, the mutation operation guides population evolution by using difference information generated by different individuals.

Many researchers have recently introduced various mutation strategies to further enhance the algorithm's performance. such as DE/rand/1, DE/best/1 [14], DE/current-to-rand/1, and DE/current-to-pbest/1 [15]. These algorithms have greatly improved the performance of DE. For example, DE/rand/1 can thoroughly explore the entire search space, while DE/best/1 can converge quickly based on superior solutions. The DE/current-to-pbest/1 algorithm proposed by Zhang and Sanderson strikes a good balance between exploration and convergence speed, and it has found widespread use in subsequent research [16, 17]. Based on the advantages of different mutation operations in solving various problems, some researchers have combined multiple mutation strategies to achieve certain effects [18–21]. In summary, the choice of mutation strategy directly determines the exploratory and exploitative performance of individuals. However, the impact of mutation strategies on the determination of individual stagnation remains an unexplored area.

Parameter settings play a crucial role in algorithm performance. In the DE algorithm, the main parameters include the scaling factor (F), crossover rate (CR), and population size (NP). The scaling factor F and the crossover rate CR primarily control the magnitude of disturbances during the evolutionary process, which significantly impacts the algorithm's exploratory and exploitative capabilities. Traditionally, F and CR are often set to fixed values. However, to better adapt to diverse optimization problems, researchers have begun to explore adaptive or dynamic adjustment strategies based on the characteristics of the problem. For example, dynamically adjusting the F value based on feedback information during the evolutionary process can achieve better performance balance during the execution of the algorithm [22–24]. Additionally, researchers are also exploring fixed or adaptive population size strategies to accommodate the population's needs at different stages of iteration, aiming to balance the diversity and convergence of the population [25–29]. In summary, by conducting in-depth studies on parameter settings, researchers aim to enhance the population's search performance. However, as far as the author is aware, there are currently no studies on using parameter settings to improve the algorithm's performance in detecting local optima.

In DE algorithms, the selection strategy has a significant influence on algorithm performance. Most existing DE algorithms adopt a greedy selection strategy based on fitness values. However, some researchers attempt to enhance algorithm performance by modifying the selection strategy in various ways [30, 31]. For instance, Das et al [32] introduced a novel selection strategy, which calculates the likelihood of an individual accepting an inferior trial vector by considering the fitness value ratio between the target vector and the trial vector. This approach can prevent population stagnation but overlooks the impact of the iteration phase, which may affect population evolution, the process where individuals in the population gradually approach the optimal solution. Moreover, Yu et al. [33] introduced a survivor selection method inspired by the simulated annealing algorithm. This approach calculates the survival probability of the trial vector based on factors such as the temperature and the fitness difference between the trial vector and the parent vector. Abbas et al [34]. introduced a DE algorithm based on tournament selection. The tournament selection strategy is more conducive to enhancing the diversity of the population during the selection process. However, each evaluation criterion only considers the fitness value of the individual, which may lead to inefficient searching. Ghosh et al. [35]. determine the survival probability of failed trial vectors based on the fitness difference between the trial vector and the target vector, as well as the Manhattan distance between them. Despite its ability to dynamically fine-tune the exploration and exploitation capabilities of the population, it does not consider the impact of individual update status, which may also affect the evolution of the population. Finally, Zeng et al [36], based on the aforementioned considerations, proposed a new selection strategy based on evolutionary status. Unlike previous methods, this approach does not use fitness as the criterion for accepting worse individuals but adjusts the probability of accepting discarded trial vectors using individual update states and iteration states. This method bolsters the population's capability to escape local optima, boosting variety in the initial phases and promoting convergence in subsequent stages. In summary, although some researchers have conducted studies on selection strategy, the current research in this area remains insufficient.

Based on the foregoing discussion, we have identified that selection strategy that accept discarded individuals with inferior fitness under certain conditions can enhance the performance of the DE algorithm. The existing research on selection strategy primarily falls into three categories:

1. Acceptance probability based on fitness or Euclidean distance differences between trial and parent vectors: This method enhances the diversity of the population and prevents evolutionary stagnation from accepting overly inferior trial individuals. However, it entirely neglects the individual's inherent potential for exploration and exploitation, thus failing to implement targeted selection strategies for individuals.
2. Optimal individual selection from subsets of generated trial individuals and the original population: This approach transforms the comparison between trial individuals and target individuals into a comparison among all individuals within subsets, allowing the entire population to select the most fitness-optimized indi-

viduals from these subsets. Nevertheless, this strategy solely relies on fitness values for selection, which could lead to a rapid decline in population diversity. Moreover, it overlooks the individuals' potential for exploration and exploitation, thereby affecting further optimization of individuals.

3. Selection strategy based on iteration phases and individual update status:: Not only does this method consider the update status of individuals, enhancing the algorithm's capability to escape local optima, but it also adjusts the focus on exploration and exploitation based on the iteration phase. However, it overlooks the individuals' exploration and exploitation potential, hindering the enhancement of these capabilities in the population. Furthermore, the acceptance of inferior individuals is solely based on the quality of trial individuals generated in the current iteration phase, and such randomness impedes the individuals' rapid escape from local optima.

In this paper, a fitness-distance-based selection (FDS) strategy is proposed to address the above issues. In particular, individuals are categorized to utilize different selection strategies. Individuals with exploration potential utilize a selection strategy that considers the Euclidean distance between the target vector and trial vectors, while individuals with higher exploitation potential employ a fitness-based selection strategy. This approach ensures that each individual chooses the most appropriate selection strategy based on their own characteristics. Moreover, this paper for the first time introduces a novel approach, namely improving the scaling factor (F), to enhance the selection strategy's ability to accept discarded trial individuals and escape local optima. At last, this paper presents a new DE variant, called fitness-distance-based DE (FDDE). Numerous experimental tests were conducted based on the CEC 2017 benchmark set [37], the CEC 2022 benchmark set [38], and the CEC 2011 benchmark set [39], and the FDDE algorithm is compared with six state-of-the-art DE algorithms and four recent and competitive evolutionary algorithms. The experimental findings indicate that FDDE surpasses the other algorithms in terms of performance. The primary key points of this paper are as follows:

1. A new selection strategy, FDS, is proposed, which implements different selection ways for individuals with exploration potential and exploitation potential. This strategy leverages the concept that discarded trial vectors, although not immediately successful, still contain valuable information that can be harnessed to enhance the evolutionary process. The theoretical basis of the FDS strategy is rooted in the premise that these discarded vectors are not merely failures but are a source of untapped potential that, if properly integrated, can provide critical insights into the solution space.
2. A novel scaling factor control method is introduced within the mutation strategy to assist the FDS strategy in detecting stagnation during local searches, consequently enhancing the algorithm's performance. Moreover, the theoretical basis of this proposed scaling factor control method lies in its ability to adjust the search range of the mutation strategy. This adjustment optimizes the balance between

exploration during population evolution and the detection of local optima, thereby significantly improving the algorithm's effectiveness.
3. The FDS strategy and the new scaling factor control formula are applied to the DISH algorithm [40], and a novel DE variant, FDDE, is introduced, which has been proved effective through a series of benchmark tests.

The rest of this paper is structured as follows: Sect. 2 provides an introduction to the DE algorithm and its various variants. Section 3 introduces the motivation and specific working principles of the FDS strategy, as well as the new scaling factor F control formula, and proposes the FDDE algorithm based on this. Section 4 conducts comprehensive experiments using the proposed algorithm and conducts a detailed analysis of the experimental results, comparing them with six state-of-the-art DE algorithm variants and four competitive evolutionary algorithms. Lastly, Sect. 5 summarizes the paper and discusses future research directions.

2 Related work

In this section, we will review the relevant literature on the DE algorithm and provide detailed introductions to several DE variants that are related to this paper.

2.1 Preliminary knowledge

DE algorithm is an efficient and versatile population-based optimization algorithm, renowned for its simplicity and robustness. It operates through mechanisms of mutation, crossover, and selection to iteratively improve a population of candidate solutions. In DE, new candidate solutions are generated by adding the weighted difference between several population individuals to a base member. This process creates a diverse pool of potential solutions, and through crossover and selection procedures, the crossover strategy combines elements from parent vectors to create trial vectors, introducing diversity into the population. The selection strategy in DE compares trial vectors with their corresponding target vectors, retaining only those that provide a fitness improvement. The mutation strategy of DE/current-to-pbest/1 proposed by Tanabe has been proved to be effective. Moreover, most DE algorithms still employ binomial crossover and greedy selection strategies, as illustrated specifically in Eqs. (1), (2), (3).

$$V_i^g = X_i^g + F \times \left(X_{pbest}^g - X_i^g\right) + F \times \left(X_{r1}^g - \overline{X_{ra}^g}\right) \tag{1}$$

$$U_{i,j}^g = \begin{cases} V_{i,j}^g & \text{if rand}(0,1) \leq CR \text{ or } j = j_{\text{rand}} \\ X_{i,j}^g & \text{otherwise} \end{cases} \tag{2}$$

$$X_i^{g+1} = \begin{cases} X_i^g, & \text{if } f(X_i^g) < f(U_i^g) \\ U_i^g, & \text{otherwise} \end{cases} \tag{3}$$

where V_i^g is the mutation vector of the target vector X_i^g in the gth generation, U_i^g is the trial vector. X_{pbest}^g denotes the best individual among the top $p\%$ individuals in the gth generation. ra is a randomly generated integer from the set $1, 2, \ldots, NP + |A|$, where A represents the archive of parent vectors that have failed in the selection strategy. $\overline{X}$ is the union of the archive A and the population. The indices r_1 are randomly generated integers from the set $\{1, 2, \ldots, NP\}$, and they are all distinct from each other. rand(0,1) is a uniformly distributed random number within the interval [0,1], D represents the dimension of the problem, j_{rand} is a randomly generated integer ranging from [1, D], and CR is a crossover rate that spans from 0 to 1. f(x) represents the value of the objective function for solution x.

Current improvements in the DE algorithm focus primarily on three areas: mutation strategy, parameter adjustments, and selection strategy. In terms of mutation strategy, research predominantly utilizes two approaches: employing a single mutation strategy and combining multiple mutation strategies. For example, the strategy DE/current-to-pbest/1 proposed by Tanabe achieves a good balance between exploration and exploitation within the population. Other studies aim to merge mutation strategies with strong exploratory capabilities with those that excel in exploitation to enhance the overall evolutionary capacity of the population [41]. Regarding parameter improvements, these include adjustments to the scaling factor, crossover rate, and population size. Currently, improvements to scaling factors and crossover rates mainly utilize two strategies: a population-based adaptive parameter strategy and an individual-based adaptive parameter strategy. For instance, Zhu [42] proposed a multipopulation DE algorithm with dynamic parameter settings for each population, while Brest and Greiner [43] introduced the JDE algorithm, which employs adaptive parameter settings to adjust the population parameters F and CR. Qin [44] introduced an adaptive parameter control DE algorithm, SaDE, which records the parameters of superior individuals in each generation during the iteration process and uses these parameters to generate offspring parameters. Meng [45] proposed a method using wavelet basis functions to generate the scaling factor F for each individual. Research on population size has shown that dynamically adjusting the size can significantly enhance the performance of the DE algorithm. For example, Tanabe and others proposed a new strategy, LPSR [46], based on the SHADE algorithm [47], which controls the evolutionary process by linearly reducing the population size. Under this strategy, individuals with the worst fitness values in the population are discarded. Based on this strategy, they introduced a new type of DE mutation called LSHADE. To meet the needs of different evolutionary stages for population size, Zhang [48] proposed a DE algorithm with nonlinear population size reduction, AGDE-MPP. Zeng [49] proposed a method of periodically adding and removing individuals from the population, which improves the population's ability to escape local optima and enhances convergence. In terms of selection strategy improvements, recent studies have shown that selection strategies can significantly enhance the performance of the DE algorithm. Current research is mainly divided into three categories: 1) strategies based on the differences between trial individuals and original

individuals, which increase population diversity; 2) merging all trial individuals with target individuals into a subset and selecting superior individuals within the subset to proceed to the next generation, thereby accelerating population convergence; and 3) selection strategies based on the state of individuals, accepting suboptimal solutions to help stagnant individuals escape local optima. These strategies attempt to fully utilize the information of discarded trial individuals to enhance the performance of the DE algorithm.

2.2 Success-history-based DE variants

2.2.1 Success-history-based DE

SHADE [47] is a further improvement based on JADE algorithm [15]. SHADE introduces a new historical memory, including M_{CR} and M_F. This archive retains successful CR and F values from past evolutionary stages, and it influences the derivation of subsequent crossover rates and scaling factors by referencing this stored data. Tanabe and Fukunaga proposed a linear population size reduction (LPSR) method, which adjusts the population size based on the number of fitness evaluations as a criterion. The specific adjustment formula is as follows:

$$N_G = N^{init} + \left(\frac{N^{min} - N^{init}}{maxnfes} \right) \times fes \tag{4}$$

where N_G represents the population size of the G generation, N^{init} represents the initial population size, N^{min} represents the minimum population size, fes represents the number of fitness evaluations, and maxnfes represents the maximum number of fitness evaluations. When $N_{G+1} < N_G$, the $(N_G - N_{G+1})$ individuals with the worst fitness values will be removed from the population. Based on the LSPR strategy and the SHADE algorithm, LSHADE algorithm was proposed.

Based on LSHADE, the techniques for regulating the scaling factor and crossover rate were further enhanced by Brest et al. [50], leading to the proposal of iLSHADE. At the same time, iLSHADE also calculates the p in Eq. (7) which is used to generate the X^g_{pbest}. The upper bound of p, p_{max}, is set to 0.25, and the lower bound of p, p_{min}, is set to half of p_{max}. Moreover, based on iLSHADE, Brest [51] improved Eq. (1), introduced a new mutation strategy DE/current-to-pbest-w/1, and proposed a new algorithm called jSO. The specific mutation strategy formula is shown in Eq. (5), and the calculation of F_w is shown in Eq. (6).

$$V^g_i = X^g_i + F_w \times (X^g_{\text{pbest}} - X^g_i) + F \times (X^g_{r1} - \overline{X^g_{ra}}) \tag{5}$$

$$F_w = \begin{cases} 0.7 \times F, & \text{if } fes < 0.2 \times maxnfes \\ 0.8 \times F, & \text{else if } fes < 0.4 \times maxnfes \\ 1.2 \times F, & \text{otherwise} \end{cases} \tag{6}$$

$$p = p_{min} + \frac{fes}{maxnfes} \times (p_{max} - p_{min}) \tag{7}$$

In order to enhance the exploration ability of the population in high-dimensional spaces, Viktorin [40] introduced distance-based parameter adaptation for SHADE and subsequently presented a DE variant known as DISH. The DISH algorithm improves the original weight updating formulas of the scaling factor and crossover rate. Instead of computing weights w_i from the fitness difference between the trial vector and the parent vector, the DISH algorithm calculates weights based on their Euclidean distance. The specific formula is depicted in Eq. (8). The generated weight is then applied to Eq. (9) and Eq. (10) to generate new scaling factors and crossover rates.

$$w_i = \frac{\sqrt{\sum_{j=1}^{D}(X_{i,j}^{g} - u_{i,j}^{g})^2}}{\sum_{k=1}^{|S_{CR}|}\sqrt{\sum_{j=1}^{D}(X_{k,j}^{g} - u_{k,j}^{g})^2}} \tag{8}$$

$$\begin{cases} M_{CR} = \text{mean}_{WL}(S_{CR}) \\ \text{mean}_{WL}(S_{CR}) = \frac{\sum_{K=1}^{|S_{CR}|} W_k \times S_{CR,k}^2}{\sum_{K=1}^{|S_{CR}|} W_k \times S_{CR,k}} \end{cases} \tag{9}$$

$$\begin{cases} M_F = \text{mean}_{WL}(S_F) \\ \text{mean}_{WL}(S_F) = \frac{\sum_{K=1}^{|S_F|} W_k \times S_{F,k}^2}{\sum_{K=1}^{|S_F|} W_k \times S_{F,k}} \end{cases} \tag{10}$$

In this context, $mean_{WL}$ denotes the weighted lehmer mean, where w_i represents the weight corresponding to individual i. Additionally, S_{CR} and S_F denote the sets containing values of crossover rate and scaling factor, respectively.

2.2.2 ESDE

The ESDE algorithm introduces a novel selection strategy (ESS) aimed at assisting individuals in breaking free from local optima. By adopting specific strategies, the algorithm can accept discarded trial vectors in certain situations. However, blindly accepting suboptimal solutions in any situation is infeasible, as it may lead to individual stagnation. Therefore, the ESS strategy considers two key factors when accepting inferior solutions: 1) the update status of the individual [52–54] and 2) the evolutionary stage of the individual. The update status of the individual reflects the probability of the individual being trapped in a local optimum. The more the generations an individual stops updating, the higher the chance it is stuck in a local optimum. The iteration stage is used to control when to accept inferior individuals. In the early stages of iteration, the algorithm considers accepting inferior trial vectors,

while in the later stages, in order to accelerate convergence, the algorithm does not accept trial vectors worse than the target vector.

As discussed above, ESDE allows individuals to accept inferior solutions with a certain probability, which enhances their ability to escape local optima. As shown in Eq. (11), α signifies the upper limit of accepting inferior individuals, β regulates the influence of an individual's non-update count on *Ac_rate*, *stop_gen* denotes the consecutive generations without target vector updates, γ governs the effect of function evaluation count on *Ac_rate*,and maxnfes represents the predetermined maximum number of function evaluations.

$$Ac\_rate = \alpha \times \frac{1}{1 + exp(\beta - stop\_gen)} \times \frac{1}{1 + exp(fes - \gamma \times maxnfes)} \tag{11}$$

3 Fitness-distance-based DE algorithm

3.1 Motivation

Greedy selection operators have been extensively studied and implemented in single-objective DE algorithms. Despite their widespread use, these strategies often lead individuals to become trapped in local optima. Studies on the ESDE algorithm reveal that it probabilistically accepts discarded trial vectors based on both the individual's update status and the current stage of iteration, assisting individuals in escaping local optima. However, ESDE does not thoroughly assess how the acceptance of these discarded vectors influences the individuals' ability to escape local optima. This oversight could lead to algorithmic instability or a reduction in exploration within promising regions, thereby diminishing the algorithm's overall performance. Furthermore, since individuals exhibit varying levels of exploration and exploitation capabilities within the search space, tailored selection strategies should be applied when accepting inferior individuals.

In traditional DE algorithms and their variants, the setting of the scaling factor F is primarily aimed at facilitating a balance between exploration and exploitation for individuals. However, research on the impact of the scaling factor on individual local optimum detection remains very limited, and this lack of study could lead to inaccuracies in the determination of local optima. Given this research gap, this study aims to deeply analyze the nature of individual local optimum detection and, based on this, specifically set a reasonable scaling factor to effectively balance exploration and local optimum detection during the evolutionary process. Detailed analysis and methodology will be introduced in Sect. 3.2.

To address the above issues, a fitness-distance-based selection strategy and a new scaling factor control are proposed. With this approach, we expect to select trial vectors based on their characteristics, so as to enhance the algorithm's ability to escape local optima, and reduce the impact of misjudgment on population evolution. Based

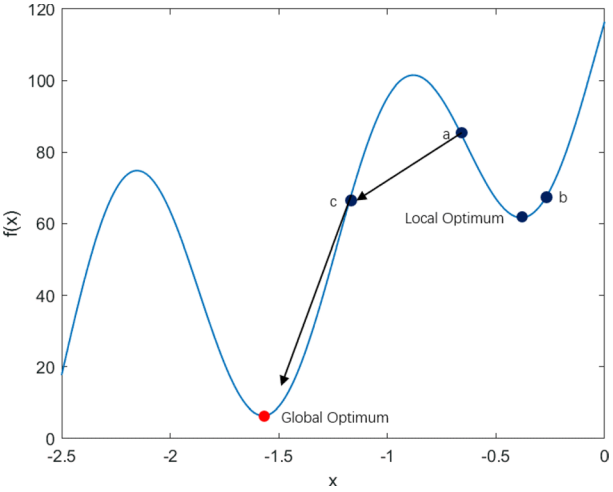

Fig. 1 Mechanism of FDS strategy in helping exploration layer individuals escape local optima

on the above strategies, a new DE variant, FDDE is proposed. We will provide a comprehensive overview of the FDDE algorithm in the subsequent sections.

3.2 Fitness-distance-based selection strategy

In our proposed FDDE, the algorithm performs an ascending fitness-based sorting of the population. The top 100*elite_rate*% of individuals are considered as exploitation layer individuals with higher exploitation potential, while the remaining individuals are considered as exploration layer individuals with higher exploration potential. The algorithm initially employs Eq. (11) to compute the acceptance probability of an individual for a discarded trial vector. When an individual is determined to accept a discarded vector, then the algorithm employs the fitness-distance-based selection (FDS) strategy.

In the exploration layer, individuals with lower fitness are more inclined to explore new areas. Once these individuals encounter local optima, it indicates that their surrounding area lacks potential for further exploration, making it crucial for them to quickly move away from these regions to seek new possibilities. As illustrated in Fig. 1, in such situations, traditional greedy selection strategies often fail to propel individuals forward in their evolutionary journey. Consider an individual that has generated trial vector points a and b in the last two generations; if we only consider the most recent trial vector, we might be inclined to choose vector b, which is more similar to the original individual. However, this similarity could lead to stagnation. In contrast, selecting the trial vector a, which bears less resemblance to the original individual, can more effectively help the individual escape from local optima and explore new domains. This approach could evolve vector a into vector c, achieving breakthroughs that vector b could not. Not only does this strategy aid individuals in escaping local optima, but it also fosters exploration of unknown areas, thereby enhancing the diversity of the population. Since the distance between individuals often reflects the similarity of their fitness landscapes, we propose that when individuals

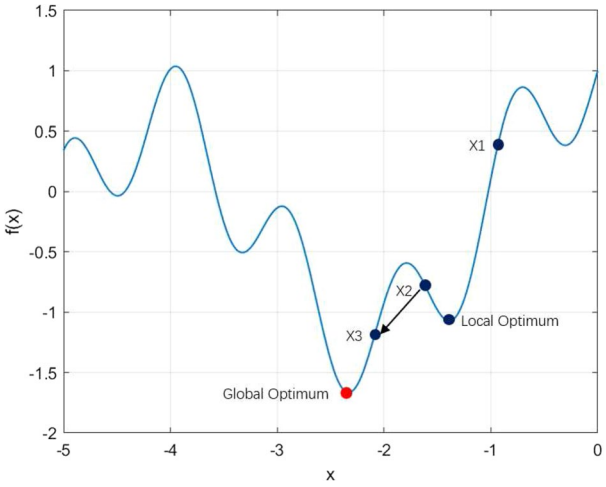

Fig. 2 Mechanism of FDS strategy in helping exploitation layer individuals escape local optima

in the exploration layer need to accept inferior trial individuals, they should select the trial individual from the past K generations that is farthest from the original individual in terms of Euclidean distance.

In the exploitation layer, since its individuals already has better fitness, accepting too inferior individuals will hardly help it find better solutions. Therefore, while ensuring that the algorithm does not accept too inferior trial vectors, the individual accepts slightly worse individuals with a certain probability. However, it is difficult to define what constitutes "too inferior" individuals. As shown in Fig. 2, after an individual falls into a local optimum, the individual in the stagnant stage may generate two trial vectors: X2 and X1. However, due to the inferior fitness value of vector X1, the individual will choose to accept vector X2, which is more likely to be perturbed to individual X3, thus helping the individual escape the local optimum and find a better solution. It is worth noting that when misjudgment occurs, this method of selecting trial vectors based on fitness can also ensure that individuals stay in better areas; as a result, it preserves the information of promising regions. thereby reducing the impact of misjudgment to some extent. By employing this approach, not only can the individual's ability to escape local optima be enhanced, but it also improves the exploitation capability of elite individuals within the population. In this study, $k = 2$ is adopted based on a series of experiments.

In this paper, a fitness-distance-based DE algorithm is proposed. This algorithm divides the population into an exploration layer and an exploitation layer to enhance the ability of different individuals to break out of local optima. A larger exploitation layer is advantageous for the population to exploit the excellent regions, while a larger exploration layer is beneficial for the population to explore more areas. And as the iterations progress, the population will gradually transition from exploration to exploitation. Therefore, we need to dynamically adjust the size of the layers. As mentioned previously, the value of *elite_rate* influences the size of layers. To better

meet the population's needs for exploration and exploitation at different stages, the formula for calculating *elite_rate* is as follows:

$$elite\_rate = \frac{nfes}{\gamma \times maxnfes} \tag{12}$$

In the above formula, nfes represents the current number of function evaluations, while maxnfes represents the maximum number of function evaluations. The setting of γ is consistent with the setting in Eq. (11).

3.3 Improved parameter settings

Existing studies often overlook the crucial role of parameters in optimizing the performance of selection strategies. At the heart of the selection strategy is the accurate determination of individual stagnation, which relies on tracking changes in an individual's fitness over a series of iterations. If an individual does not show an improvement in fitness within a certain number of iterations, it is considered to have stagnated. However, this method is highly dependent on the search area within the mutation strategy. If the individual's mutation strategy fails to explore around the evolving individual or if the search step is too large, it is fundamentally impossible to effectively determine whether the individual is in a local optimum area. The DE/current-to-pbest/1 mutation strategy used in this paper can effectively explore the area around an individual, yet inappropriate search step sizes may also lead to inaccuracies in the detection of individual stagnation.

To minimize the chances of incorrectly identifying individuals ensnared in local optima, we introduce a new approach to controlling the scaling factor (F) by imposing restrictions on excessively large scaling factors, and both the search capability of individuals and the stability of the mutation strategy around individuals are ensured. This restriction allows for an appropriate mutation step size and prevents excessive fluctuations or jumps. As a result, the accuracy of determining whether an individual is stagnant based on the number of stagnation instances is improved. Therefore, restricting excessively large scaling factors helps ensure the accuracy of stagnation detection, thereby enhancing the performance of the DE algorithm. The formula is shown in Eq. (13).

$$F = \begin{cases} \min(F, 0.6) & \text{if } nfes \leq 0.6 \times maxnfes \\ F & \text{otherwise} \end{cases} \tag{13}$$

3.4 Framework of FDDE

A new DE variant FDDE is proposed, which is based on FDS and the new scaling factor. The pseudocode of FDDE is shown in Algorithm 1.

From Algorithm 1, there are four differences between DISH and FDDE: (1) K and count values are initialized in line 3; (2) a new scaling factor formula is used in lines 22–24; (3) sorting of the population and calculation of *elite_rate* are done in lines 28–29; and (4) the fitness-distance-based selection strategy is introduced in lines 35–41.

Algorithm 1 FDDE algorithm

Set NP $= 6 \cdot D + 43 \cdot \log(D) + 15$, $N_{\min} = 4$, $p_{\min} = 0.125$, $p_{\max} = 0.25$, $p = p_{\max}$
Initialize p_0 randomly according to Eq. (1), $p_0 = X_1, X_2, ..., X_{NP}$
Set $M_F = 0.5$, $M_{CR} = 0.8$, $A = \emptyset$, $K = 2$, $fes = NP$, Set all values in count equal to 0, H = 5
while $fes <$ maxnfes **do**
  $S_F = \emptyset$, $S_{CR} = \emptyset$, $CR = \emptyset$, $F = \emptyset$
  for $i = 1$ to NP **do**
    $r =$ randomly select from [1,H]
    if $r == H$ **then**
      $M_F^r = 0.9$, $M_{CR}^r = 0.9$
    end if
    if $M_{CR}^r < 0$ **then**
      $CR_i = 0$
    else
      $CR_i = N(M_{CR}^r, 0.1)$
    end if
    if $nfes < 0.25 \cdot$ maxnfes **then**
      $CR_i = \max(CR_i, 0.7)$
    else if $nfes < 0.5 \cdot$ maxnfes **then**
      $CR_i = \max(CR_i, 0.6)$
    end if
    $F_i = \mathcal{C}(M_{F,r}, 0.1)$
    if $nfes < 0.6 \cdot$ maxnfes *and* $F_i \geq 0.6$ **then**
      $F_i = 0.6$
    end if
    Compute V_i^g by mutation *Eq.* (5)
    Compute u_i^g by crossover *Eq.* (2)
  end for
  Sort the population by fitness values ascend
  Calculate *elite_rate* according to *Eq.* (12)
  for $i = 1$ to NP **do**
    if $f(u_i^g) < f(x_i^g)$ **then**
      $x_i^{g+1} = u_i^g$, $count(i) = 0$
      $A \Leftarrow x_i^g$, $S_F \Leftarrow F_i$, $CR_i \Leftarrow S_{CR}$
    else
      $count(i) = count(i) + 1$
      Calculate *Ac_rate* according to *Eq.* (11)
      if $i < elite\_rate \cdot NP$ *and* rand$(0,1) <$ *ac_rate* **then**
        $x_i^g =$ the discarded vector with the farthest Euclidean distance of individual i within the past K generations
      else if $i \geq elite\_rate \cdot NP$ *and* rand$(0,1) <$ *ac_rate* **then**
        $x_i^g =$ the discarded vector with the best fitness of individual i within the past K generations.
      end if
    end if
    Update A if necessary
  end for
  Calculate NP_{new} according to *Eq.* (4)
  $fes = fes + NP$
  $NP = NP_{\text{new}}$
  if $S_F \neq \varnothing$ **then**
    Calculate W_k according to Eq. (8)
    Update $M_{F,k}$ and $M_{CR,k}$ according to Eq. (9) and Eq. (10)
    Update p according to Eq. (7)
  end if
end while

Table 1 Complexity of FDDE

Dimension	Algorithm	T_0	T_1	$\bar{T}_2$	$(\bar{T}_2 - T_1)/T_0$
10D	DISH	0.018	0.522	0.675	8.500
	ESDE			0.652	7.111
	FDDE			0.781	14.389
30D	DISH		0.675	0.744	3.833
	ESDE			0.826	8.389
	FDDE			0.914	13.278
50D	DISH		0.923	1.072	8.278
	ESDE			1.147	12.444
	FDDE			1.335	22.889
100D	DISH		2.038	2.859	45.611
	ESDE			2.921	49.056
	FDDE			2.948	50.556

4 Experiments and discussions

4.1 Experiment environment

In this section, to assess the FDDE algorithm's performance, this study delves into the proposed FDS strategy and the new scaling factor setting, and further tests the effectiveness of the algorithm on CEC 2017, CEC 2022, and CEC 2011.

CEC 2017 consists of 30 benchmark functions that cover various types of optimization problems, including unimodal functions (F1-F3), simple multiobjective functions (F4-F10), hybrid functions (F11-F20), and composite functions (F21-F30).

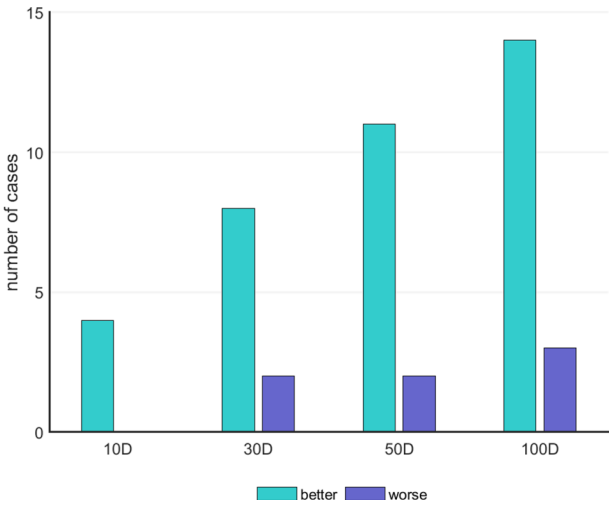

Fig. 3 Comparison between FDDE and FDDE-1

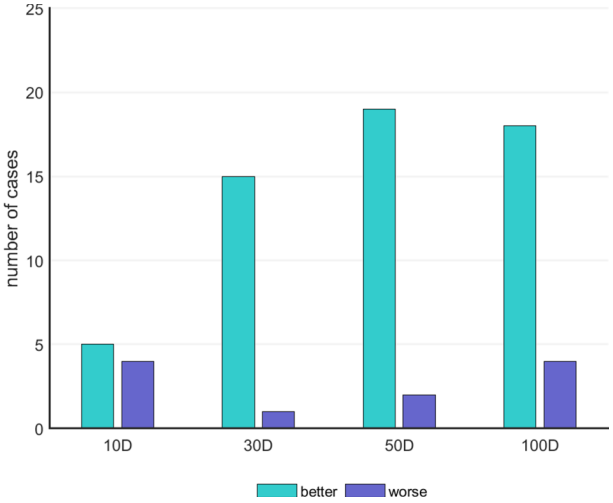

Fig. 4 Comparison between FDDE and FDDE-2

Table 2 Average rankings among FDDE algorithm variants according to the Friedman test

Algorithm	10D	30D	50D	100D
FDDE-K1	2.1500	2.4000	2.4500	2.2700
FDDE-K2	**1.6300**	**1.7200**	**1.6500**	**1.6700**
FDDE-K3	2.2200	3.9833	1.9000	2.0700

The best-ranked algorithm is emphasized in bold

Each test function includes four instances corresponding to 10D, 30D, 50D, and 100D, totaling 120 instances. Detailed information about CEC 2017 can be found in reference [37]. The optimal values for all test functions are predetermined, and the evaluation criterion is based on the disparity between the values achieved by each algorithm and the established optima. The mean value and standard deviation of the function error values are denoted as Mean and Std, respectively. In the CEC 2017 benchmark set, we set the maximum number of algorithm evaluations to 10,000 × D, and the algorithm runs 51 times on each instance, thereby avoiding inaccuracies due to algorithmic randomness. For each algorithm, if the error value falls below 10^{-8}, it is adjusted to 0.

The CEC 2022 test suite includes 12 test functions, covering four types of functions: unimodal functions (F1), basic functions (F2-F5), hybrid functions (F6-F8), and composition functions (F9-F12). In this experiment, we set the dimension of the CEC 2022 benchmark test suite to 20, tested each algorithm on each function 30 times, and took the average. Detailed information about the CEC 2022 test suite can be found in reference [38].

In the CEC 2011 benchmark set suite, seven problems are selected for testing to demonstrate the superiority of the FDDE algorithm in handling real-world problems.

The maximum number of evaluations was set to 150,000, and for each instance, 25 runs were conducted for every algorithm. The specific settings are described in reference [39]. To ensure the effectiveness of the algorithm performance, this study utilized the Wilcoxon rank-sum test, with a significance level set at 0.05.

4.2 Analysis of algorithmic complexity

The complexity of the algorithm in this paper is determined according to the method in reference [37]. The specific steps are as follows: Firstly, the test code in reference [37] is run to obtain the time T_0; secondly, the function 18 in the CEC 2017 test set is evaluated 200,000 times to obtain the time T_1; and lastly, the test algorithm is evaluated 200,000 times in function 18 and repeated five times to calculate the average value as time $\bar{T}_2$. The time complexity is then evaluated by the formula $(\bar{T}_2 - T_1) / T_0$.

Table 1 indicates that the FDDE algorithm's complexity closely resembles that of ESDE and DISH. Notably, the complexity of the FDDE algorithm does not significantly increase with the increase in dimension. All time complexity tests are executed on the following computer configuration: The programming language is MATLAB; it is compiled with MATLAB 2021; CPU is AMD Ryzen 7 5800 H with Radeon Graphics 3.20 GHz; and RAM is 16 G.

4.3 Ablation experiment

To verify the effectiveness of the FDS strategy and the new scaling factor control formula, we compared the following three methods: (1) FDDE algorithm; (2) FDDE algorithm without using the new scaling factor control formula (FDDE-1); and (3) FDDE algorithm without using the FDS strategy (FDDE-2). Figures 3 and 4 show the performance differences between FDDE, FDDE-1, and FDDE-2. The outcomes reveal that FDDE outperforms FDDE-1 across all scales (10D, 30D, 50D, and 100D), demonstrating that the FDS strategy can significantly enhance algorithm performance, especially in the 30D, 50D, and 100D. Figure 3 further illustrates the improvement of algorithm performance brought by the new scaling factor. It can be observed that the new scaling factor brings significant performance improvement in all dimensions (10D, 30D, 50D, and 100D).

Table 3 Average rankings among FDDE algorithm variants according to the Friedman test

Algorithm	10D	30D	50D	100D	Average
FDDE	**2.5333**	2.7667	2.4667	2.4000	**2.5417**
FDDE-P2	3.2833	2.7333	2.6667	2.5333	2.8042
FDDE-P3	3.3500	**2.4833**	**2.0000**	2.5000	2.5833
ESDE	2.9333	3.3333	3.8000	3.2833	3.3375
DISH	2.9000	3.6833	4.0667	4.2833	3.7333

The best-ranked algorithm is emphasized in bold

4.4 Parameter analysis

4.4.1 Sensitivity of K values

In this paper, we propose a selection strategy based on fitness distance, where the parameter K plays a significant role in enabling individuals to specifically accept inferior trial individuals to escape local optima. To better investigate the setting of K, we employ the Friedman test to rank the performance of the FDDE algorithm under the settings of K=1, K=2, and K=3. The FDDE variants corresponding to these settings are FDDE-K1, FDDE-K2, and FDDE-K3, as detailed in Table 2.

As shown in Table 2, when comparing the three settings of K values, the setting of K=2 demonstrated the best performance across all four dimensions, while K=1 showed the least desirable effect. This indicates that relying solely on trial individuals generated in the current generation of an individual might limit the capability to overcome local optima. In the comparison between K=3 and K=2, selecting individuals based on the furthest Euclidean distance or the optimal fitness distance did not fully leverage the potential of the FDS strategy. Nevertheless, the performance under K=3 still surpasses that of the K=1 setting.

4.4.2 Sensitivity of population size

Population size has a significant impact on the performance of the DE algorithm. Recognizing the popularity of various population settings, this study compares the FDDE algorithm described in this paper with two FDDE variants that employ widely used initial population sizes. These two variants are: FDDE-P2, which sets the population size to $18 \times D$, and FDDE-P3, which determines the population size based on the formula $25 \times log(D) \times sqrt(D)$. We use the statistical method Friedman test to rank these algorithms in order to assess the impact of different population size settings on performance.

As shown in Table 3, the parameters set in this paper achieved the best ranking in both 10D and 100D dimensions. Additionally, the FDDE algorithm demonstrated superior performance across these four dimensions, reflecting the advantages of this algorithm in terms of population size settings. Furthermore, FDDE-P3 achieved the best ranking in the 30D and 50D dimensions. Notably, on the 30D, 50D, and 100D dimensions, the FDDE algorithm and its variants were among the top three, showing that even with three different population size setting strategies, the FDDE algorithm consistently outperformed the DISH and ESDE algorithms in all scenarios. These results not only highlight the excellent performance of the FDDE algorithm but also demonstrate its flexibility and robustness in adjusting population size parameters. The performance of the FDDE algorithm further confirms its low sensitivity to different population sizes, allowing it to maintain stable high performance across a wide range of applications.

4.5 Comparison results

4.5.1 Comparison results with six recent DE variants on CEC 2017

In this section, we compare the performance of the FDDE algorithm with six state-of-the-art DE algorithms, including JADE [15], LSHADE [46], jSO [51], DISH [40], ESDE [36], and HIP-DE [55]. To faithfully represent the performance of the original algorithms, the parameter settings for each algorithm are kept the same as those in the corresponding literature. As shown in Table 4, for the JADE algorithm, the population size NP is set to 100 for 10D, 120 for 30D, 150 for 50D, and 200 for 100D. The α in ESDE and FDDE is set to 0.5, β is set to 48 for 10D and 30D, and 32 for 50D and 100D, and γ is set to 0.4 for 10D and 30D, and 0.5 for 50D and 100D. The parameter settings in CEC 2011 are the same as those in the 30D setting.

Table 5 presents the performance of the FDDE algorithm in comparison with other algorithms on the CEC 2017 test set, and statistical analysis was conducted on the results of each algorithm independently run 51 times on each function, based on the Wilcoxon rank-sum test at a significance level of 0.05. The specific information of the detailed data, including the mean error values and standard deviations obtained through each algorithm, is provided in Tables 16, 17, 18, and 19. The symbols "+" (FDDE performs significantly better), "–" (FDDE performs significantly worse), and "=" (FDDE and the comparison algorithm do not show a significant difference) indicate whether the performance of the FDDE algorithm is significantly different from that of the other algorithms based on the Wilcoxon rank-sum test on the mean error values. The integer corresponding to "+,—,=" represents the number of functions that are significantly better, significantly worse, or not significantly different between the FDDE algorithm and the compared algorithm on the test set based on the statistical analysis.

Table 5 shows that FDDE is significantly better than other advanced DE algorithms. the performance difference between FDDE and JADE was the largest. FDDE has improved in 85 instances and only significantly underperformed in 9 instances when compared with JADE. When compared with LSHADE, 77 instances showed significant improvement, while only 11 instances significantly underperformed. In comparison with jSO, FDDE achieved significant improvement in 68 instances, with only 7 instances significantly underperformed. We paid particular attention to the performance of FDDE against ESDE and DISH. When compared to DISH, FDDE showed significant improvement in 63 instances and significantly underperformed in 11 instances, the improvement rate is 52.5% and the degradation rate is 9.2%. FDDE performed significantly better than ESDE in 50 instances, with only 5 instances significantly underperformed, the improvement rate is 41.7% and the degradation rate is only 4.2%. It is evident that, compared to these two algorithms, FDDE has achieved substantial improvement and fewer instances of degradation. Of note, FDDE shows significant improvements compared to both ESDE and DISH, with very few functions where FDDE significantly underperformed. the HIP-DE algorithm is the closest to FDDE. FDDE performed significantly better than HIP-DE in 68 instances, with 23 instances significantly underperformed.

Table 4 Parameter settings

Algorithm	Year	Parameters' initial settings
JADE	2009	N = 100,c = 0.01, p = 0.5, CR = 0.5, F = 0.5
LSHADE	2014	$N_{max} = 18 \times D, N_{min} = 4, H = 6, r^{arc} = 2.6, M_F = 0.5, M_{CR} = 0.5$
jSO	2017	$N_{max} = 25 \times log(D) \times sqrt(D), H = 5, M_F = 0.3, M_{CR} = 0.8, r^{arc} = 2.6,$ $p_{min} = 0.125, p_{max} = 0.25$
DISH	2019	$N_{max} = 25 \times log(D) \times sqrt(D), H = 5, M_F = 0.5, M_{CR} = 0.8, r^{arc} = 2.6,$ $p_{min} = 0.125, p_{max} = 0.25$
HIP-DE	2021	$N_{max} = 15 \times D, N_{min} = 4, M_F = 0.6, M_{CR} = 0.8, \tau_F = \tau_{CR} = 0.9, K = 6,$ $r^{arc} = 5, p_{min} = 0.05, p_{max} = 0.2$
ESDE	2022	$N_{max} = 6 \times D + 43 \times log(D) + 15, N_{min} = 4, H = 5, M_F = 0.5, M_{CR} = 0.8$
FDDE	-	$N_{max} = 6 \times D + 43 \times log(D) + 15, N_{min} = 4, H = 5, M_F = 0.5, M_{CR} = 0.8,$ K = 2

The results in Tables 6, 7, 8, and 9 are derived from statistical analysis using the Wilcoxon rank-sum test at a significance level of 0.05. As shown in Tables 6, 7, 8, and 9, in comparison with other algorithms in different dimensions and types of functions, we have observed that our method has varying degrees of advantages

Table 5 Comparison of FDDE and six other DE variants

	Metric	10D	30D	50D	100D	Total
JADE	+	14	22	24	25	85
	–	5	2	1	1	9
	=	11	6	5	4	26
LSHADE	+	9	21	22	25	77
	–	6	1	2	2	11
	=	15	8	6	3	32
jSO	+	6	17	21	24	68
	–	2	1	2	2	7
	=	22	12	7	4	45
DISH	+	8	17	20	18	63
	–	2	2	3	4	11
	=	20	11	7	8	46
ESDE	+	6	13	18	13	50
	–	0	2	2	1	5
	=	24	15	10	16	65
HIP-DE	+	6	16	20	26	68
	–	12	6	4	1	23
	=	12	8	6	3	29
Total	+	49	106	125	131	411
	–	27	14	14	11	66
	=	104	60	41	38	243

Table 6 Comparison of FDDE and six other DE variants under different function types for 10D

Algorithm	Metric	Unimodal functions	Multimodal functions	Hybrid functions	Composition functions
JADE	+	0	4	7	3
	–	0	0	1	4
	=	3	3	2	3
LSHADE	+	0	4	3	2
	–	0	0	4	2
	=	3	3	3	6
jSO	+	0	4	1	1
	–	0	0	1	1
	=	3	3	8	8
DISH	+	0	4	2	2
	–	0	0	2	0
	=	3	3	6	8
ESDE	+	0	3	2	1
	–	0	0	0	0
	=	3	4	8	9
HIP-DE	+	0	3	2	1
	–	0	0	7	5
	=	3	4	1	4
Total	+	0	22	17	10
	–	0	0	15	12
	=	18	20	28	38

in different types of functions. For 10D, specifically, our algorithm shows significant enhancements in multimodal functions with 22 instances of improvements and no instances showing a decline. Moreover, in unimodal functions, all methods in our experiments converge to the global optimum. Additionally, our algorithm demonstrates similar performance compared to other algorithms. For 30D, our algorithm achieves substantial improvements in multimodal functions, hybrid functions, and composition functions. In multimodal functions, 23 instances show significant improvements without any instances showing a decline. In hybrid functions, 49 instances exhibit significant enhancements, while 2 instances show significant deterioration. For composition functions, 33 instances demonstrate significant improvements, while 7 instances show significant deterioration. For 50D, our algorithm achieves significant improvements in multimodal functions and hybrid functions with 31 instances and 53 instances of improvement, respectively, without any instances showing significant deterioration. Moreover, in composition functions, our algorithm shows significant improvements in 42 instances and significant deteriorations in 12 instances. For 100D, our algorithm achieves significant improvements in multimodal functions and composition functions with 29 instances and 45 instances of significant enhancements, respectively, without any instances showing

Table 7 Comparison of FDDE and six other DE variants under different function types for 30D

Algorithm	Metric	Unimodal functions	Multimodal functions	Hybrid functions	Composition functions
JADE	+	1	4	10	7
	–	0	0	0	1
	=	2	3	0	2
LSHADE	+	0	3	10	8
	–	0	0	0	0
	=	3	4	0	2
jSO	+	0	4	8	5
	–	0	0	0	1
	=	3	3	2	4
DISH	+	0	5	6	6
	–	0	0	1	1
	=	3	2	3	3
ESDE	+	0	4	6	3
	–	0	0	1	1
	=	3	3	3	6
HIP-DE	+	0	3	9	4
	–	0	0	0	3
	=	3	4	1	3
Total	+	1	23	49	33
	–	0	0	2	7
	=	17	19	9	20

a significantly decline. In hybrid functions, there are 49 instances of significant improvements and 2 instances of significant deteriorations. In unimodal functions, 8 instances demonstrate significant improvements, while 4 instances show significant deteriorations. It is noted that our algorithm shows varying degrees of improvement over ESDE and DISH algorithms in all dimensions and function types.

To delve deeper into the analysis of the FDDE algorithm's performance, we conducted an analysis of the average error values for each algorithm using the Friedman test. The obtained average rankings are presented in Table 10, where smaller average ranks indicate better algorithm performance and the best ranking is highlighted in bold. Our algorithm achieved the best rankings in dimensions 30, 50, and 100. For 30D, ESDE ranks second, and DISH ranks fourth. In particular, for 50D, the FDDE algorithm exhibited a significant lead over the second-ranked ESDE algorithm, ESDE ranks second, and DISH ranks third. Furthermore, for 100D, our algorithm attained the lowest average rank among the four dimensions, ESDE ranks second, and DISH ranks third. For 10D, our algorithm ranked third, and HIP-DE ranks first. In comparison with the DISH and ESDE algorithms, our algorithm demonstrated improved performance in all dimensions.

The convergence curves can reflect the convergence characteristics of the algorithm. In this study, an analysis was conducted on some functions in 30D, 50D,

Table 8 Comparison of FDDE and six other DE variants under different function types for 50D

Algorithm	Metric	Unimodal functions	Multimodal functions	Hybrid functions	Composition functions
JADE	+	1	5	10	8
	–	0	0	0	0
	=	2	2	0	2
LSHADE	+	0	6	10	6
	–	0	0	0	2
	=	3	1	0	2
jSO	+	0	6	9	6
	–	0	0	0	2
	=	3	1	1	2
DISH	+	0	5	8	7
	–	0	0	0	3
	=	3	2	2	0
ESDE	+	0	5	6	7
	–	0	0	0	2
	=	3	2	4	1
HIP-DE	+	0	4	10	6
	–	0	0	0	3
	=	3	3	0	1
Total	+	1	31	53	40
	–	0	0	0	12
	=	17	11	7	8

and 100D on CEC 2017. F10 was selected as the test function from the simple multimodal functions, F20 was chosen as the test function for hybrid functions, and F26 was selected for composition functions. The convergence curves are shown in Figs. 5, 6, and 7. It can be observed that the proposed algorithm in this study significantly outperforms all advanced DE algorithms. In comparison with DISH and ESDE, we observed that FDDE achieved a similar convergence rate in the early stages and was able to converge to a better value in the later stages. This is because when individuals with exploitation potential get trapped in local optima, the FDDE algorithm continues to explore the regions with exploitation potential by accepting the fitness values of discarded trial individuals that have been filtered, which is advantageous for finding better solutions in the early stage. In the later stage, due to the thorough exploration in the early stage that identifies regions with development potential, the FDDE algorithm still maintains good convergence speed. Therefore, it can be observed that the FDDE achieves a good balance between exploration and exploitation. In summary, the research found that in 10D functions, this paper does not significantly differ from other algorithms, but in high-dimensional test functions such as 30D, 50D, and 100D, this paper performs excellently, outperforming the other six comparison algorithms. Upon further study, it was found that in multimodal functions, the FDDE

Table 9 Comparison of FDDE and six other DE variants under different function types for 100D

Algorithm	Metric	Unimodal functions	Multimodal functions	Hybrid functions	Composition functions
JADE	+	2	5	9	9
	–	0	0	0	0
	=	1	2	1	1
LSHADE	+	1	6	10	8
	–	1	0	0	0
	=	1	1	0	2
jSO	+	1	5	10	8
	–	1	0	0	0
	=	1	2	0	2
DISH	+	1	5	6	6
	–	1	0	1	0
	=	1	2	3	4
ESDE	+	2	2	4	5
	–	0	0	1	0
	=	1	5	5	5
HIP-DE	+	1	6	10	9
	–	1	0	0	0
	=	1	1	0	1
Total	+	8	29	49	45
	–	4	0	2	0
	=	6	13	9	15

algorithm achieved significant improvement in 10D, 30D, 50D, and 100D, with no instances of degradation. In hybrid functions and composition functions, the number of instances in which the FDDE algorithm improved and underperformed in 10D was fairly close, but in 30D, 50D, and 100D, significant improvement was achieved, significantly outperforming the six comparison algorithms.

To demonstrate the algorithm's robustness, applicability, and scalability, we utilized four distinct types of functions from the CEC2017 benchmark for our analysis. The FDDE algorithm's performance was showcased across a variety of problems using box plots. Specifically, we selected function F3 for unimodal challenges, F10 for simple multimodal problems, and functions F13 and F17 for hybrid scenarios, as well as functions F22 and F26 for composite problems. These functions were chosen because they present significant optimization challenges, providing a rigorous test of the algorithm's ability to manage uncertainty and sustain performance.

Box plots are statistical charts that illustrate data distribution effectively. They provide a five-number summary: the minimum, first quartile (Q1), median (Q2), third quartile (Q3), and maximum. The central box represents the interquartile range, with the median marked by a central line. The boundaries of the box, Q3 and Q1, delineate the upper and lower quartiles, respectively. In this study, outliers are

Table 10 Average rankings between FDDE and six other DE variants according to the Friedman test

Algorithm	10D	30D	50D	100D
FDDE	3.7500	**2.5500**	**2.1500**	**1.8667**
ESDE	3.8500	2.8500	3.2500	2.3833
DISH	4.0833	3.9833	3.8667	3.8833
jSO	3.7167	4.2000	4.0500	4.0167
LSHADE	4.3167	4.6667	4.0333	4.4833
JADE	5.0333	5.9167	6.2833	6.2167
HIP-DE	**3.2500**	3.8333	4.3667	5.1500

The best-ranked algorithm is emphasized in bold

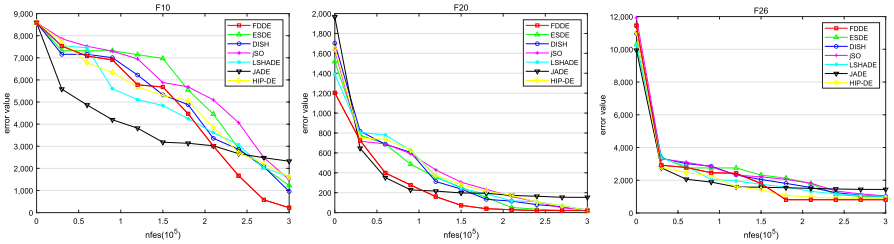

Fig. 5 Convergence curves of the mean fitness on certain test functions in 30D

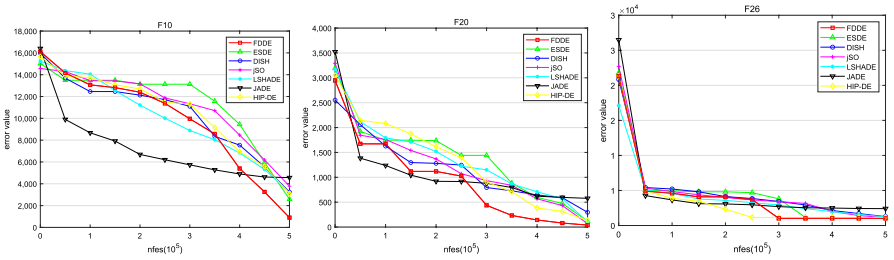

Fig. 6 Convergence curves of the mean fitness on certain test functions in 100D

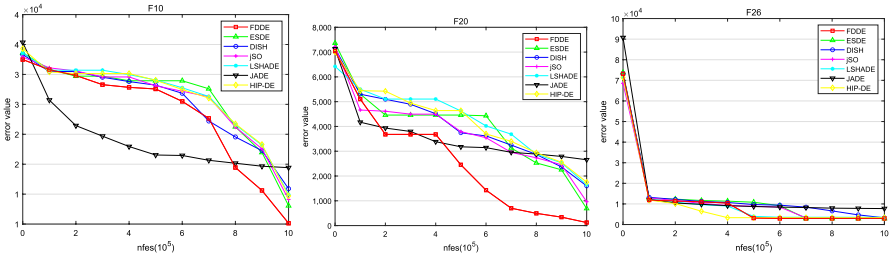

Fig. 7 Convergence curves of the mean fitness on certain test functions in 100D

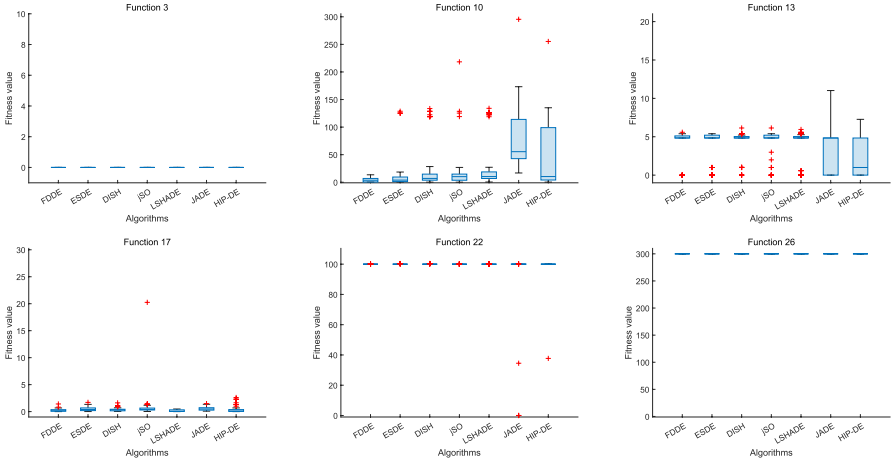

Fig. 8 Boxplot results for different function types in 10D

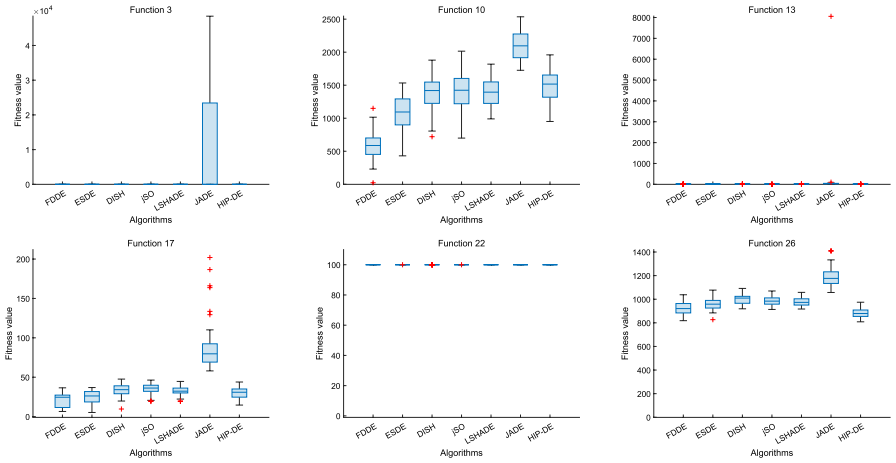

Fig. 9 Boxplot results for different function types in 30D

indicated with a red "+" to emphasize deviations, ensuring clarity in data interpretation and analysis.

As shown in Figs. 8, 9, 10, and 11, for unimodal functions, most algorithms stably converged to a satisfactory value across 10D, 30D, 50D, and 100D, except for JADE, which displayed numerous outliers and poorer convergence in 30D, 50D, and 100D. For simple multimodal problems, which feature many local optima, FDDE significantly outperformed other algorithms in terms of median values and demonstrated higher stability (smaller interquartile range). In hybrid functions, FDDE showed superior stability and convergence performance compared to others, notably in 100D where FDDE had more outliers but still outperformed other algorithms.

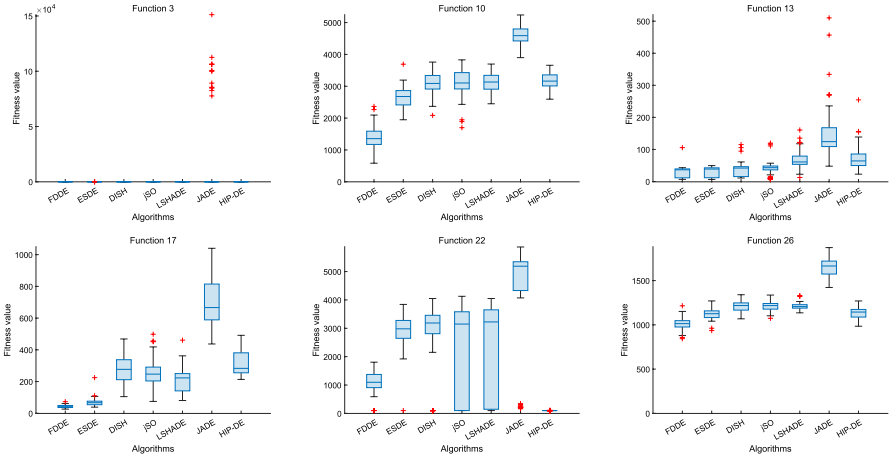

Fig. 10 Boxplot results for different function types in 50D

The presence of numerous outliers suggests that while FDDE can navigate local optima in high-dimensional settings, its improvement still depends on the exploratory scope of the population, and escaping local optima does not guarantee finding better solutions. Composite functions combine multiple function types, and here, FDDE not only maintained high stability and convergence but also showed significantly better performance than other algorithms, which struggled with convergence. This highlights FDDE's strong global search capability and effective convergence in scenarios with multiple local optima and extensive suboptimal regions. In summary, FDDE exhibits robust performance across various problem types, maintaining stability through multiple runs. This further attests to FDDE's excellent applicability and scalability across a diverse range of function types and dimensions.

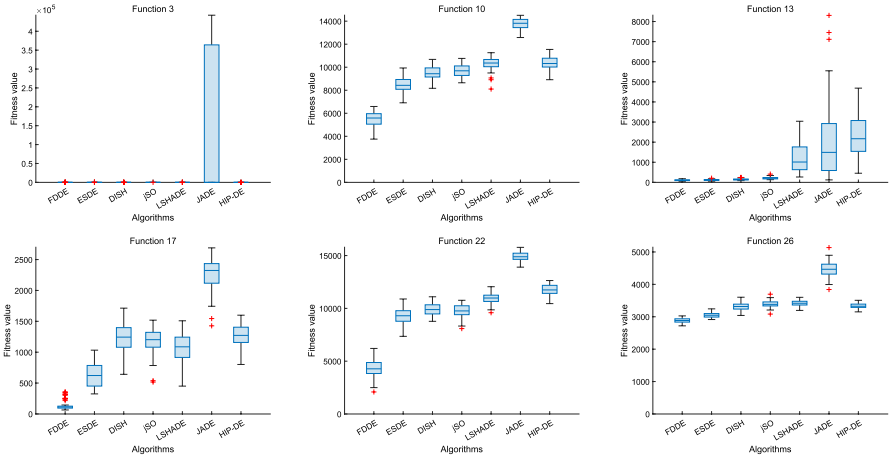

Fig. 11 Boxplot results for different function types in 100D

Table 11 Parameter settings of four advanced evolutionary algorithms

Algorithm	Year	Parameters' initial settings
PSO-sono	2022	$N = 100, iw \in \{0, 0.4, 0.9\}, \varepsilon = \frac{D}{N} \times 0.01, r = 0.5$
LEA	2024	$N = 50, h_{max} = 0.7, h_{min} = 0, \lambda_c = 0.5, \lambda_p = 0.5$
LPO	2024	$N = 30, N_e = 5$
WOA	2016	$N = 50$

4.5.2 Comparison results with four famous evolutionary algorithms on CEC 2017

In this section, we compare the proposed FDDE algorithm with four of the latest advanced evolutionary algorithms: the PSO-sono [56], LEA [57], LPO [58], and WOA algorithms [59]. The parameter settings for these four algorithms are the same as those in the original texts, as detailed in Table 11. PSO-sono represents the latest variant of the particle swarm optimization algorithm, while LEA, LPO, and WOA are innovative meta-heuristic algorithms. For each algorithm, the mean and standard deviation of fitness error values are recorded after 51 independent runs on each function. The recorded data are then subjected to statistical analysis and comparison using the Wilcoxon rank-sum test with a significance level set at 0.05.

In this paper, the experimental results presented in Table 12 demonstrate a comprehensive performance enhancement of our proposed FDDE algorithm when compared with four advanced evolutionary algorithms: PSO-sono, LEA, WOA, and LPO. Specifically, in the 10D dimension, FDDE showed significant improvements in 22, 26, 30, and 21 test functions respectively against PSO-sono, LEA, WOA, and LPO, but significantly underperformed in 4, 4, and 1 test problems respectively. In higher dimensions of 30D, 50D, and 100D, FDDE's performance was exceptionally strong, significantly outperforming all test functions compared to PSO-sono, LEA, and WOA. Compared to LPO, FDDE was significantly better in 27, 28, and 29 functions in these dimensions, with only 1, 0, and 0 functions where it underperformed respectively.

The analysis of the data indicates that the FDDE algorithm has made significant progress in all tested dimensions. Although the improvements at 10D were relatively modest, enhancements in the dimensions of 30D, 50D, and 100D were markedly greater. This phenomenon reflects the increased complexity of problem-solving as the number of dimensions increases, leading to more local optima. Thanks to the FDS strategy in the FDDE algorithm, which is particularly suited for selectively discarding inferior solutions, the algorithm exhibits robustness against multiple local optima, making FDDE more effective at handling high-dimensional problems.

4.5.3 Comparison results on CEC 2022

To further explore the performance and robustness of the FDDE algorithm across different function types, we continued to use the CEC 2022 test suite to test the algorithm. The characteristics of the functions in the CEC 2022 test suite include nonlinearity, multimodality, non-convexity, non-differentiability, and high complexity, which impose

Table 12 Comparison of FDDE and other evolutionary algorithms

Algorithm	Metric	10D	30D	50D	100D
PSO-sono	+	22	30	30	30
	–	4	0	0	0
	=	4	0	0	0
LEA	+	26	30	30	30
	–	4	0	0	0
	=	0	0	0	0
WOA	+	30	30	30	30
	–	0	0	0	0
	=	0	0	0	0
LPO	+	21	27	28	29
	–	1	0	0	1
	=	8	3	2	0

high demands on the algorithm's performance. The parameter settings of the comparison algorithms are the same as those used in the CEC 2017 30D test set. As shown in Table 13, "±/=" represents significant improvement, significant underperformance, or no significant difference based on the Wilcoxon rank-sum test compared to other algorithms. "Rank" indicates the average ranking obtained from the Friedman test.

Unimodal functions are used to assess the convergence capability of algorithms. In comparison with advanced algorithms such as ESDE, DISH, jSO, LSHADE, and HIP-DE, the FDDE algorithm proposed in this paper demonstrated outstanding performance, ranking first, while the JADE algorithm ranked last. This result clearly demonstrates FDDE's capability for rapid and effective convergence.

For basic functions, which are characterized by their multimodality, there is a high demand for the algorithm's exploration ability. The FDDE algorithm ranked second in this category. According to the Wilcoxon rank-sum test results, FDDE performed comparably to the ESDE algorithm. Compared to the DISH and jSO algorithms, FDDE was significantly better in 1 instance and showed no significant difference in 3 instances. Against the JADE and HIP-DE algorithms, FDDE was significantly better in two instances and showed no significant difference in two instances. These findings indicate that the FDDE algorithm can effectively explore extensive search spaces and accurately locate potential optimization areas.

Hybrid functions are created by combining various basic test functions to mimic the diversity and complexity of real-world problems and demand more from the adaptability and robustness of optimization algorithms. Based on Friedman test results, the FDDE algorithm ranked first, significantly outperforming other algorithms. According to the Wilcoxon rank-sum test, compared to the ESDE algorithm, FDDE performed significantly better in 2 instances but underperformed in one. When compared with DISH, jSO, and LSHADE, FDDE was significantly better in one instance and showed no significant difference in three. In comparison with JADE and HIP-DE, FDDE was significantly better in 3 instances. Overall, the FDDE algorithm maintained robust performance when dealing with hybrid functions,

Table 13 Comparison of algorithm performance on different function types according to Wilcoxon rank-sum test and Friedman test rankings

Types of Functions	Index	FDDE	ESDE	DISH	jSO	LSHADE	JADE	HIP-DE
Unimodal function	±/=	–	0/0/1	0/0/1	0/0/1	0/0/1	1/0/0	0/0/1
	Rank	3.50	3.50	3.50	3.50	3.50	7.00	3.50
Basic functions	±/=	–	0/0/4	1/0/3	1/0/3	2/0/2	2/0/2	2/0/2
	Rank	3.50	2.75	3.75	4.00	4.50	5.50	4.00
Hybrid functions	±/=	–	2/1/0	2/0/1	2/0/1	2/0/1	3/0/0	3/0/0
	Rank	1.33	3.00	3.67	4.00	5.00	7.00	4.00
Composition functions	±/=	–	1/0/3	1/0/3	1/0/3	1/0/3	2/1/1	1/0/3
	Rank	3.25	4.50	3.25	3.50	3.75	5.00	4.75
Overall	±/=	–	3/1/8	4/0/8	4/0/8	5/0/7	8/1/3	6/0/6
	Rank	2.87	3.46	3.54	3.79	4.29	5.83	4.21

showing a stable improvement over other DE variants, with only a few instances of poor performance. This indicates that the fitness-distance selection strategy can reliably assist the algorithm in exploring the entire search space.

Composition functions combine multiple different test functions into a single optimization problem. Each basic function in the composite maintains its independence and impacts different regions of the entire search space. These functions often have multiple local optima. According to the results of the Friedman test, the FDDE

Table 14 Description of the real-world problems

Problem	Dimension	introduction
problem1	6	Parameter Estimation for Frequency Modulated (FM)
problem2	30	Lennard–Jones Potential Problem
problem5	30	Tersoff Potential for model Si (B)
problem6	30	Tersoff Potential for model Si (B)
problem7	20	Spread Spectrum Radar Polly phase Code Design
problem12	26	Messenger:Space craft Trajectory Optimization Problem
problem13	22	Cassini 2:Spacecraft Trajectory Optimization Problem

Table 15 Comparison results for the real-world problems

Problem	JADE	LSHADE	jSO	DISH	ESDE	HIP-DE	FDDE
problem1	2.07E−01	4.34E−04	1.36E+00	1.22E+00	8.13E−01	7.25E−05	0.00E+00
	(+)	(+)	(=)	(=)	(=)	(+)	
problem2	− 2.30E+01	− 2.61E+01	− 2.57E+01	− 2.58E+01	− 2.63E+01	− 2.63E+01	− 2.63E+01
	(+)	(=)	(+)	(+)	(=)	(=)	
problem5	− 3.54E+01	− 3.64E+01	− 3.60E+01	− 3.62E+01	− 3.60E+01	− 3.64E+01	− 3.62E+01
	(+)	(=)	(=)	(=)	(=)	(=)	
problem6	− 2.90E+01	− 2.92E+01	− 2.91E+01	− 2.92E+01	− 2.87E+01	− 2.91E+01	− 2.90E+01
	(−)	(−)	(−)	(−)	(+)	(=)	
problem7	1.17E+00	1.15E+00	1.14E+00	1.15E+00	1.05E+00	1.17E+00	1.03E+00
	(+)	(+)	(+)	(+)	(=)	(=)	
problem12	1.74E+01	1.54E+01	1.51E+01	1.50E+01	1.50E+01	1.78E+01	1.50E+01
	(+)	(+)	(=)	(=)	(=)	(+)	
problem13	1.75E+01	1.33E+01	1.33E+01	1.38E+01	1.40E+01	1.40E+01	1.46E+01
	(+)	(=)	(=)	(=)	(=)	(=)	
Total ±/=	6/1/0	3/1/3	2/1/4	2/1/4	1/0/5	2/0/4	

and DISH algorithms ranked joint first. Based on the Wilcoxon rank-sum test results, compared to the ESDE, DISH, jSO, LSHADE, and HIP-DE algorithms, FDDE performed significantly better on one instance and showed no significant differences on 3 instances. Compared to the JADE algorithm, FDDE performed significantly better on two instances but was significantly worse on one, demonstrating that FDDE's superior selection strategy and improved scaling factor parameters enhance the algorithm's performance in handling local optima challenges.

In comprehensive comparisons across all function types, the FDDE algorithm ranked first and maintained significant performance improvements compared to six other excellent DE variants. Overall, the FDDE algorithm demonstrated outstanding exploration and exploitation capabilities, maintained stability in handling complex problems with multiple local optima, and showed high robustness across tests of various function types.

4.5.4 Comparison results on CEC 2011

To validate the effectiveness of FDDE in practical problems, CEC 2011 was selected as the testing platform to evaluate FDDE and six comparative DE variants algorithms. Seven bounded function problems were chosen. The specific problem description and dimension settings are shown in Table 14. The parameter configurations for each algorithm are consistent with their settings in the 30-dimensional space within CEC 2017.

Table 15 shows that FDDE exhibits significant performance differences compared to the JADE algorithm. In six out of the seven problems, FDDE outperforms JADE algorithm, with only one problem showing slightly inferior performance.

When compared to DISH and ESDE algorithms, FDDE shows similar performance, surpassing DISH algorithm in two problems while being slightly inferior in one, and outperforming ESDE algorithm in one problem. These results indicate the potential of the FDDE algorithm in solving real-world problems, with its overall performance being comparable to or better than advanced DE algorithms.

5 Conclusion

In this paper, we proposed a novel fitness-distance-based selection (FDS) strategy and introduce a new scaling factor formula. Specifically, FDS selects and accepts appropriate discarded trial vectors based on individual exploration and exploitation potential under certain conditions. Furthermore, the adjusted scaling factor enhances the identification of individual stagnation in the FDS strategy, leading to the improvement of the algorithm performance. The experimental results demonstrate that FDS significantly enhances algorithm performance, and the new scaling factor further improves the algorithm's effectiveness. When compared with six advanced DE algorithms, the proposed fitness-distance-based DE (FDDE) algorithm shows remarkable superiority from CEC 2017, CEC 2022, and CEC 2011, and FDDE shows great potential in real-world optimization problems. Furthermore, when FDDE is compared against four other advanced evolutionary algorithms on the CEC 2017 benchmark, the results consistently reveal its significant advantages over competing algorithms.

While this study has contributed valuable insights into the application of differential evolution strategies, it also uncovers several limitations that point towards avenues for future research:

1. The study primarily uses fitness values to assess individual potential for exploration and exploitation. This straightforward approach does not consistently ensure precise identification, especially in complex scenarios where fitness landscapes are deceptive.
2. The feasibility of the proposed selection strategy in other evolutionary algorithms remains to be fully explored. The results of this study are promising, but the applicability of the strategy across different evolutionary frameworks could further validate its effectiveness.
3. The current method for detecting local optima requires further refinement to improve its accuracy. As the detection of local optima is crucial for the timely adjustment of search strategies in evolutionary algorithms, enhancing the precision of this detection mechanism could significantly impact the overall performance of the algorithm.

Appendix

See Tables 16, 17, 18 and 19.

Table 16 Mean and Std of algorithms under 10D of CEC 2017

	FDDE		ESDE		DISH		jSO		LSHADE		JADE		HIP-DE	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
F1	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00
F2	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00
F3	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00
F4	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00
F5	**1.01E+00**	8.56E−01	1.64E+00	8.40E−01	2.40E+00	7.22E−01	2.24E+00	7.67E−01	2.54E+00	8.52E−01	3.27E+00	8.42E−01	1.84E+00	8.53E−01
F6	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00
F7	**1.16E+01**	5.29E−01	1.21E+01	8.42E−01	1.24E+01	7.36E−01	1.21E+01	6.58E−01	1.23E+01	5.73E−01	1.36E+01	7.68E−01	1.19E+01	7.55E−01
F8	**1.23E+00**	8.10E−01	2.03E+00	8.89E−01	2.44E+00	8.75E−01	2.15E+00	8.53E−01	2.68E+00	8.07E−01	3.65E+00	8.28E−01	2.01E+00	9.22E−01
F9	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00
F10	**3.97E+00**	3.43E+00	1.26E+01	2.91E+01	2.40E+01	4.09E+01	1.93E+01	4.01E+01	2.76E+01	4.29E+01	7.97E+01	5.35E+01	4.13E+01	5.92E+01
F11	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	4.61E−01	7.15E−01	2.55E+00	6.88E−01	5.96E−01	8.58E−01
F12	2.01E+01	4.34E+01	**3.88E−01**	1.56E−01	5.09E+00	2.35E+01	1.27E+01	3.78E+01	4.95E+01	6.47E+01	9.23E+01	8.39E+01	1.65E+01	4.13E+01
F13	4.39E+00	1.63E+00	3.89E+00	2.11E+00	4.46E+00	1.49E+00	4.27E+00	1.71E+00	4.33E+00	1.70E+00	3.68E+00	2.53E+00	**2.24E+00**	2.55E+00
F14	**1.95E−02**	1.39E−01	1.17E−01	3.24E−01	3.90E−02	1.95E−01	1.17E−01	3.80E−01	1.68E−01	3.49E−01	5.33E−01	3.74E−01	9.83E−01	9.99E−01
F15	2.71E−01	2.21E−01	2.49E−01	2.25E−01	2.64E−01	2.22E−01	2.48E−01	2.17E−01	**1.31E−01**	1.72E−01	4.62E−01	1.61E−01	1.89E−01	2.15E−01
F16	8.12E−01	1.61E+00	5.66E−01	3.23E−01	3.93E−01	2.58E−01	5.44E−01	2.71E−01	**3.47E−01**	1.54E−01	1.45E+00	8.05E−01	4.36E−01	2.16E−01
F17	2.45E−01	2.58E−01	5.11E−01	4.03E−01	3.88E−01	3.16E−01	8.86E−01	2.79E+00	**1.35E−01**	1.50E−01	5.52E−01	3.37E−01	3.85E−01	6.23E−01
F18	2.61E−01	2.10E−01	2.53E−01	2.04E−01	2.75E−01	2.12E−01	2.72E−01	2.03E−01	2.60E−01	2.00E−01	1.21E+00	3.89E+00	**2.01E−01**	2.08E−01
F19	9.57E−03	1.32E−02	1.22E−02	1.23E−02	1.25E−02	1.14E−02	1.08E−02	1.31E−02	1.32E−02	1.11E−02	4.87E−02	2.22E−02	**7.55E−03**	1.12E−02
F20	4.79E−01	1.96E−01	7.84E−01	2.79E+00	3.43E−01	1.29E−01	3.61E−01	1.31E−01	**0.00E+00**	0.00E+00	1.22E−02	6.12E−02	**0.00E+00**	0.00E+00
F21	1.84E+02	3.65E+01	1.76E+02	4.38E+01	1.59E+02	5.18E+01	**1.38E+02**	5.05E+01	1.62E+02	5.11E+01	1.67E+02	4.75E+01	1.53E+02	5.15E+01
F22	1.00E+02	4.85E−02	1.00E+02	9.76E−02	1.00E+02	1.01E−01	**1.00E+02**	4.82E−02	1.00E+02	4.83E−02	9.48E+01	2.14E+01	9.88E+01	8.72E+00
F23	3.03E+02	1.78E+00	3.02E+02	1.92E+00	3.03E+02	1.61E+00	3.02E+02	1.96E+00	3.04E+02	1.31E+00	3.05E+02	1.28E+00	**3.01E+02**	1.55E+00
F24	3.25E+02	3.24E+01	3.30E+02	4.56E+00	3.03E+02	7.50E+01	2.81E+02	9.59E+01	3.16E+02	6.41E+01	**2.90E+02**	7.64E+01	2.92E+02	8.11E+01

Table 16 (continued)

	FDDE		ESDE		DISH		jSO		LSHADE		JADE		HIP-DE	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
F25	4.11E+02	2.10E+01	4.07E+02	1.83E+01	**4.00E+02**	8.89E+00	4.10E+02	2.00E+01	4.11E+02	2.10E+01	4.20E+02	2.32E+01	4.21E+02	2.29E+01
F26	**3.00E+02**	0.00E+00	**3.00E+02**	0.00E+00	**3.00E+02**	0.00E+00	**3.00E+02**	0.00E+00	**3.00E+02**	0.00E+00	**3.00E+02**	0.00E+00	**3.00E+02**	0.00E+00
F27	3.89E+02	1.97E−01	3.89E+02	1.88E−01	3.89E+02	1.78E−01	3.89E+02	2.13E−01	3.89E+02	1.26E−01	**3.89E+02**	5.52E−01	3.93E+02	1.59E+00
F28	4.25E+02	1.49E+02	4.07E+02	1.42E+02	4.02E+02	1.42E+02	4.11E+02	1.48E+02	3.59E+02	1.22E+02	3.73E+02	1.25E+02	**3.45E+02**	1.04E+02
F29	**2.31E+02**	2.24E+00	2.32E+02	2.26E+00	2.34E+02	2.45E+00	2.34E+02	3.27E+00	2.35E+02	3.35E+00	2.44E+02	5.78E+00	2.33E+02	4.73E+00
F30	3.24E+04	1.60E+05	6.45E+04	2.22E+05	1.64E+04	1.14E+05	6.45E+04	2.22E+05	8.05E+04	2.45E+05	4.64E+04	1.84E+05	**3.99E+02**	1.45E+01

Table 17 Mean and Std of algorithms under 30D of CEC 2017

	FDDE		ESDE		DISH		jSO		LSHADE		JADE		HIP-DE	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
F1	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00
F2	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00
F3	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00
F4	5.86E+01	2.52E−14	5.86E+01	2.61E−14	5.86E+01	2.95E−14	5.86E+01	2.83E−14	5.86E+01	3.00E−14	**4.79E+01**	2.38E+01	5.40E+01	1.59E+01
F5	**5.19E+00**	1.83E+00	6.60E+00	1.79E+00	1.08E+01	1.99E+00	8.99E+00	1.71E+00	6.60E+00	1.40E+00	2.94E+01	3.91E+00	7.89E+00	1.36E+00
F6	5.36E−08	1.75E−07	1.25E−07	3.95E−07	1.06E−07	3.47E−07	1.41E−08	4.12E−08	1.21E−08	3.75E−08	**0.00E+00**	0.00E+00	2.68E−09	1.92E−08
F7	3.70E+01	1.67E+00	3.80E+01	1.59E+00	4.13E+01	1.77E+00	3.90E+01	1.74E+00	3.76E+01	1.19E+00	5.84E+01	3.97E+00	**3.64E+01**	9.88E−01
F8	**5.66E+00**	1.36E+00	6.97E+00	1.96E+00	1.08E+01	2.30E+00	9.41E+00	1.95E+00	7.36E+00	1.62E+00	2.77E+01	4.10E+00	7.76E+00	1.20E+00
F9	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00
F10	**6.01E+02**	2.19E+02	1.07E+03	2.74E+02	1.36E+03	2.46E+02	1.40E+03	2.85E+02	1.40E+03	2.08E+02	2.09E+03	2.00E+02	1.50E+03	2.10E+02
F11	1.84E+01	2.63E+01	1.14E+01	2.00E+01	**5.62E+00**	1.15E+01	1.44E+01	2.33E+01	3.56E+01	2.83E+01	3.40E+01	2.78E+01	1.89E+01	2.49E+01
F12	2.66E+02	1.60E+02	1.45E+02	1.05E+02	**1.07E+02**	9.38E+01	3.64E+02	2.35E+02	1.18E+03	4.15E+02	1.15E+03	4.65E+02	1.16E+03	3.14E+02
F13	1.58E+01	5.77E+00	1.84E+01	2.31E+00	1.91E+01	3.56E+00	1.84E+01	4.18E+00	1.72E+01	7.14E+00	1.94E+02	1.12E+03	**1.54E+01**	6.74E+00
F14	2.06E+01	3.05E+00	2.16E+01	1.32E+00	**1.83E+01**	7.42E+00	2.14E+01	2.75E+00	2.19E+01	1.22E+00	5.81E+03	7.96E+03	2.20E+01	2.94E+00
F15	**1.11E+00**	7.16E−01	1.52E+00	9.99E−01	1.72E+00	1.21E+00	1.99E+00	1.37E+00	3.73E+00	1.72E+00	1.21E+03	3.36E+03	3.04E+00	1.91E+00
F16	**1.37E+01**	2.78E+00	1.55E+01	3.76E+00	3.35E+01	4.38E+01	3.26E+01	4.76E+01	3.57E+01	1.90E+01	4.33E+02	1.21E+02	1.24E+02	8.64E+01
F17	**2.02E+01**	8.06E+00	2.49E+01	8.43E+00	3.35E+01	7.71E+00	3.51E+01	6.83E+00	3.29E+01	5.55E+00	8.83E+01	3.17E+01	3.03E+01	7.24E+00
F18	2.08E+01	4.51E−01	2.07E+01	3.08E−01	**2.07E+01**	2.96E−01	2.08E+01	3.64E−01	2.28E+01	1.66E+00	2.49E+04	5.91E+04	2.25E+01	1.57E+00
F19	**4.00E+00**	1.19E+00	4.78E+00	1.76E+00	5.15E+00	1.65E+00	5.74E+00	2.15E+00	5.94E+00	1.80E+00	1.81E+03	4.81E+03	5.41E+00	2.09E+00
F20	2.15E+01	7.67E+00	**1.99E+01**	7.92E+00	2.82E+01	8.91E+00	2.62E+01	7.36E+00	3.06E+01	5.46E+00	1.18E+02	5.39E+01	3.71E+01	1.85E+01
F21	**2.06E+02**	2.08E+00	2.08E+02	2.06E+00	2.12E+02	2.31E+00	2.10E+02	2.05E+00	2.08E+02	1.53E+00	2.29E+02	4.33E+00	2.07E+02	1.59E+00
F22	**1.00E+02**	1.44E−14	**1.00E+02**	6.39E−14	**1.00E+02**	1.37E−13	**1.00E+02**	6.39E−14	**1.00E+02**	1.44E−14	**1.00E+02**	1.44E−14	**1.00E+02**	1.44E−14
F23	3.52E+02	4.78E+00	3.51E+02	3.98E+00	3.58E+02	4.31E+00	3.57E+02	4.50E+00	3.53E+02	3.09E+00	3.76E+02	5.76E+00	**3.44E+02**	3.87E+00
F24	4.26E+02	1.92E+00	4.28E+02	2.64E+00	4.31E+02	2.41E+00	4.31E+02	2.62E+00	4.28E+02	2.01E+00	4.42E+02	4.88E+00	**4.22E+02**	2.02E+00

Table 17 (continued)

	FDDE		ESDE		DISH		jSO		LSHADE		JADE		HIP-DE	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
F25	3.87E+02	6.15E−03	3.87E+02	5.69E−03	**3.87E+02**	5.48E−03	3.87E+02	6.41E−03	3.87E+02	2.20E−02	3.87E+02	1.29E−01	3.87E+02	2.22E−02
F26	9.22E+02	5.44E+01	9.62E+02	5.11E+01	9.98E+02	3.78E+01	9.85E+02	3.75E+01	9.80E+02	3.32E+01	1.19E+03	7.87E+01	**8.83E+02**	3.86E+01
F27	5.06E+02	6.55E+00	**5.02E+02**	6.57E+00	5.02E+02	6.28E+00	5.04E+02	5.44E+00	5.07E+02	4.67E+00	5.03E+02	7.08E+00	5.05E+02	4.86E+00
F28	3.15E+02	3.86E+01	3.13E+02	3.65E+01	**3.08E+02**	2.88E+01	3.22E+02	4.56E+01	3.58E+02	5.88E+01	3.46E+02	5.58E+01	3.23E+02	4.46E+01
F29	**4.29E+02**	1.14E+01	4.33E+02	1.06E+01	4.39E+02	1.25E+01	4.39E+02	9.00E+00	4.38E+02	7.93E+00	4.82E+02	3.26E+01	4.35E+02	5.64E+00
F30	1.98E+03	4.91E+01	1.97E+03	2.19E+01	1.97E+03	2.49E+01	**1.97E+03**	3.43E+01	2.01E+03	6.55E+01	2.12E+03	1.47E+02	2.07E+03	8.72E+01

Table 18 Mean and Std of algorithms under 50D of CEC 2017

	FDDE		ESDE		DISH		jSO		LSHADE		JADE		HIP-DE	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
F1	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00
F2	**0.00E+00**	0.00E+00	3.00E−10	2.14E−09	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00
F3	**0.00E+00**	0.00E+00	1.36E−09	7.33E−09	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	2.15E+04	4.24E+04	**0.00E+00**	0.00E+00
F4	5.76E+01	5.63E+01	6.77E+01	5.39E+01	**5.37E+01**	5.44E+01	6.69E+01	5.37E+01	6.24E+01	5.02E+01	5.79E+01	4.60E+01	8.61E+01	4.69E+01
F5	**1.03E+01**	2.49E+00	1.21E+01	2.63E+00	1.86E+01	3.19E+00	1.56E+01	3.11E+00	1.17E+01	2.35E+00	6.40E+01	8.49E+00	1.45E+01	1.88E+00
F6	1.79E−07	2.87E−07	5.97E−07	7.16E−07	4.08E−07	5.89E−07	5.64E−07	1.45E−06	8.42E−06	5.98E−05	**5.50E−08**	1.26E−07	2.29E−07	5.33E−07
F7	6.34E+01	2.25E+00	6.46E+01	2.40E+00	7.03E+01	3.81E+00	6.67E+01	2.79E+00	6.42E+01	1.97E+00	1.15E+02	7.99E+00	**6.14E+01**	1.48E+00
F8	**9.85E+00**	2.62E+00	1.10E+01	3.01E+00	1.79E+01	3.59E+00	1.65E+01	2.99E+00	1.12E+01	1.94E+00	6.63E+01	6.88E+00	1.49E+01	2.05E+00
F9	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	2.25E−01	3.03E−01	**0.00E+00**	0.00E+00
F10	**1.42E+03**	3.65E+02	2.67E+03	3.34E+02	3.09E+03	3.34E+02	3.10E+03	4.54E+02	3.12E+03	2.87E+02	4.61E+03	3.24E+02	3.17E+03	2.69E+02
F11	**2.22**E+01	3.03E+00	2.30E+01	3.25E+00	2.53E+01	3.13E+00	2.88E+01	2.46E+00	4.69E+01	1.09E+01	1.29E+02	5.90E+01	4.81E+01	7.70E+00
F12	1.72E+03	4.84E+02	**1.63E+03**	4.53E+02	1.74E+03	3.74E+02	1.87E+03	4.67E+02	2.22E+03	5.00E+02	4.93E+03	3.93E+03	2.36E+03	4.40E+02
F13	3.09E+01	1.74E+01	**3.07E+01**	1.55E+01	3.75E+01	2.31E+01	4.19E+01	2.39E+01	6.78E+01	2.90E+01	1.51E+02	8.70E+01	7.55E+01	4.18E+01
F14	**2.28E+01**	1.37E+00	2.40E+01	1.74E+00	2.42E+01	2.32E+00	2.40E+01	2.16E+00	3.16E+01	3.97E+00	1.66E+04	4.46E+04	3.25E+01	3.38E+00
F15	**2.07E+01**	1.84E+00	2.14E+01	1.92E+00	2.16E+01	2.07E+00	2.38E+01	2.49E+00	4.59E+01	1.33E+01	2.19E+02	1.02E+02	5.01E+01	1.23E+01
F16	**1.00E+02**	5.90E+01	1.16E+02	8.66E+01	4.60E+02	1.62E+02	4.35E+02	1.54E+02	3.27E+02	1.14E+02	9.10E+02	1.88E+02	3.98E+02	9.26E+01
F17	**4.32E+01**	9.17E+00	6.92E+01	2.74E+01	2.70E+02	9.91E+01	2.45E+02	1.03E+02	2.13E+02	8.22E+01	6.91E+02	1.44E+02	3.11E+02	7.41E+01
F18	2.34E+01	1.83E+00	2.30E+01	1.54E+00	**2.30E+01**	1.47E+00	2.55E+01	2.77E+00	5.53E+01	2.26E+01	3.02E+03	1.99E+04	5.48E+01	1.72E+01
F19	**9.92E+00**	2.16E+00	1.10E+01	2.35E+00	1.25E+01	2.54E+00	1.40E+01	3.11E+00	3.09E+01	1.01E+01	1.31E+02	3.77E+01	4.22E+01	1.27E+01
F20	**4.63E+01**	8.07E+00	9.11E+01	5.67E+01	1.23E+02	5.22E+01	1.19E+02	6.43E+01	1.64E+02	6.57E+01	5.10E+02	1.25E+02	1.65E+02	5.91E+01
F21	**2.12E+02**	3.30E+00	2.13E+02	3.12E+00	2.21E+02	3.34E+00	2.18E+02	3.50E+00	2.14E+02	2.35E+00	2.66E+02	7.79E+00	2.16E+02	2.45E+00
F22	1.12E+03	3.92E+02	2.87E+03	5.76E+02	2.66E+03	1.32E+03	2.15E+03	1.69E+03	1.98E+03	1.74E+03	4.12E+03	2.09E+03	**1.00E+02**	1.86E+00
F23	4.27E+02	8.19E+00	4.32E+02	7.28E+00	4.36E+02	5.77E+00	4.35E+02	5.18E+00	4.32E+02	3.76E+00	4.92E+02	1.10E+01	**4.26E+02**	7.77E+00
F24	**5.05E+02**	4.70E+00	5.08E+02	3.37E+00	5.19E+02	4.33E+00	5.18E+02	3.56E+00	5.11E+02	2.59E+00	5.45E+02	8.88E+00	5.08E+02	4.55E+00

Table 18 (continued)

	FDDE		ESDE		DISH		jSO		LSHADE		JADE		HIP-DE	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
F25	4.81E+02	3.53E+00	4.82E+02	3.82E+00	4.81E+02	2.76E+00	4.81E+02	3.27E+00	**4.81E+02**	2.33E+00	5.14E+02	3.37E+01	4.83E+02	5.23E+00
F26	**1.01E+03**	6.98E+01	1.12E+03	6.47E+01	1.21E+03	5.44E+01	1.21E+03	5.39E+01	1.21E+03	3.86E+01	1.66E+03	9.89E+01	1.13E+03	6.67E+01
F27	5.46E+02	2.55E+01	5.39E+02	2.53E+01	**5.31E+02**	2.44E+01	5.33E+02	1.74E+01	5.39E+02	1.47E+01	5.38E+02	1.97E+01	5.33E+02	7.98E+00
F28	**4.59E+02**	1.93E−13	4.59E+02	2.09E−13	4.59E+02	2.50E−13	4.59E+02	1.93E−13	4.65E+02	1.66E+01	4.87E+02	2.39E+01	4.88E+02	2.43E+01
F29	**3.33E+02**	1.32E+01	3.50E+02	1.89E+01	3.57E+02	1.55E+01	3.57E+02	1.39E+01	3.48E+02	1.25E+01	4.89E+02	5.95E+01	3.75E+02	2.38E+01
F30	6.88E+05	8.06E+04	6.49E+05	7.65E+04	6.33E+05	6.69E+04	6.49E+05	6.84E+04	6.69E+05	9.54E+04	6.84E+05	7.35E+04	**6.20E+05**	4.16E+04

Table 19 Mean and Std of algorithms under 100D of CEC 2017

	FDDE		ESDE		DISH		jSO		LSHADE		JADE		HIP-DE	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
F1	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	2.90E−10	2.07E−09	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00
F2	**3.93E+01**	2.18E+02	5.09E+04	2.88E+05	6.91E+04	2.66E+05	9.31E+03	3.18E+04	2.17E+05	1.39E+06	2.30E+04	1.46E+05	5.47E+03	3.88E+04
F3	1.95E−04	2.10E−04	2.20E+01	1.46E+01	5.29E−05	5.43E−05	4.74E−06	6.16E−06	2.65E−06	3.08E−06	1.41E+05	1.86E+05	**2.14E−07**	3.67E−07
F4	2.00E+02	1.09E+01	2.01E+02	1.03E+01	2.03E+02	1.03E+01	1.99E+02	9.09E+00	1.95E+02	2.85E+01	**8.59E+01**	6.65E+01	1.96E+02	3.38E+00
F5	1.74E+01	4.29E+00	1.77E+01	5.52E+00	3.61E+01	7.66E+00	3.54E+01	5.92E+00	2.67E+01	6.94E+00	**1.73E+02**	1.75E+01	3.88E+01	4.90E+00
F6	**4.84E−06**	3.22E−06	1.17E−05	7.80E−06	5.31E−06	4.60E−06	9.27E−05	2.47E−04	1.88E−03	1.42E−03	1.45E−04	5.51E−04	1.16E−03	1.19E−03
F7	1.26E+02	4.38E+00	**1.26E+02**	3.48E+00	1.41E+02	6.94E+00	1.40E+02	5.03E+00	1.34E+02	1.94E+01	2.86E+02	1.66E+01	1.32E+02	3.64E+00
F8	**1.69E+01**	3.94E+00	1.82E+01	6.14E+00	3.55E+01	6.74E+00	3.65E+01	5.98E+00	2.75E+01	5.84E+00	1.76E+02	1.91E+01	4.13E+01	4.77E+00
F9	**1.76E−03**	1.25E−02	7.02E−03	2.43E−02	1.76E−03	1.25E−02	0.00E+00	0.00E+00	1.81E−01	2.84E−01	1.11E+01	1.17E+01	8.84E−02	3.30E−01
F10	**5.44E+03**	6.78E+02	8.47E+03	6.55E+02	9.48E+03	5.52E+02	9.69E+03	5.46E+02	1.01E+04	1.53E+03	1.38E+04	4.56E+02	1.04E+04	5.54E+02
F11	5.69E+01	4.29E+01	**5.50E+01**	3.02E+01	5.86E+01	2.03E+01	1.15E+02	3.44E+01	3.81E+02	1.16E+02	9.66E+03	7.24E+03	4.40E+02	8.39E+01
F12	1.44E+04	6.25E+03	1.56E+04	8.58E+03	**1.42E+04**	6.69E+03	1.98E+04	1.09E+04	2.29E+04	1.01E+04	1.70E+04	8.61E+03	2.21E+04	9.92E+03
F13	**1.08E+02**	2.83E+01	1.18E+02	3.32E+01	1.44E+02	3.68E+01	2.13E+02	5.44E+01	1.21E+03	7.53E+02	2.00E+03	1.97E+03	2.27E+03	1.00E+03
F14	**3.53E+01**	4.02E+00	3.68E+01	3.66E+00	4.17E+01	4.72E+00	7.12E+01	1.18E+01	2.51E+02	4.59E+01	5.36E+02	1.15E+02	2.57E+02	2.93E+01
F15	1.26E+02	3.79E+01	**1.17E+02**	3.03E+01	1.31E+02	3.82E+01	2.07E+02	5.19E+01	2.42E+02	6.36E+01	3.45E+02	1.57E+02	2.34E+02	4.21E+01
F16	**1.08E+02**	6.90E+01	6.80E+02	3.11E+02	1.82E+03	3.22E+02	1.64E+03	2.95E+02	1.57E+03	3.17E+02	2.98E+03	3.13E+02	1.87E+03	2.41E+02
F17	**1.38E+02**	7.58E+01	6.39E+02	2.01E+02	1.23E+03	2.64E+02	1.17E+03	2.17E+02	1.04E+03	2.78E+02	2.27E+03	2.64E+02	1.27E+03	1.88E+02
F18	1.69E+02	3.67E+01	1.58E+02	4.39E+01	**1.41E+02**	3.16E+01	2.01E+02	4.22E+01	2.22E+02	5.29E+01	5.12E+02	2.47E+02	2.11E+02	4.18E+01
F19	**6.11E+01**	1.04E+01	5.40E+01	6.63E+00	6.21E+01	7.75E+00	1.43E+02	2.60E+01	1.72E+02	3.75E+01	2.32E+02	5.05E+01	1.73E+02	2.02E+01
F20	**1.44E+02**	1.30E+02	8.59E+02	3.07E+02	1.23E+03	2.34E+02	1.25E+03	2.31E+02	1.43E+03	2.77E+02	2.27E+03	2.19E+02	1.65E+03	1.87E+02
F21	**2.42E+02**	4.33E+00	2.45E+02	5.07E+00	2.62E+02	7.29E+00	2.62E+02	6.39E+00	2.52E+02	3.61E+01	3.94E+02	1.97E+01	2.68E+02	5.11E+00
F22	**4.36E+03**	9.06E+02	9.21E+03	8.04E+02	9.90E+03	5.74E+02	9.74E+03	6.59E+02	1.07E+04	1.60E+03	1.49E+04	4.81E+02	1.17E+04	5.38E+02
F23	**5.47E+02**	1.01E+01	5.58E+02	9.97E+00	5.70E+02	9.53E+00	5.66E+02	8.72E+00	5.60E+02	6.34E+00	6.81E+02	1.08E+01	6.11E+02	1.24E+01
F24	**8.90E+02**	6.16E+00	8.97E+02	5.49E+00	9.19E+02	9.01E+00	9.24E+02	9.85E+00	9.19E+02	6.88E+00	1.03E+03	1.98E+01	9.30E+02	1.48E+01

Table 19 (continued)

	FDDE		ESDE		DISH		jSO		LSHADE		JADE		HIP-DE	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
F25	**6.81E+02**	4.54E+01	6.88E+02	4.92E+01	6.84E+02	4.19E+01	7.21E+02	4.34E+01	7.50E+02	2.81E+01	7.52E+02	4.14E+01	7.40E+02	3.56E+01
F26	**2.88E+03**	7.12E+01	3.04E+03	7.28E+01	3.32E+03	1.14E+02	3.39E+03	1.09E+02	3.42E+03	9.06E+01	4.46E+03	2.53E+02	3.33E+03	8.11E+01
F27	6.24E+02	1.97E+01	**6.16E+02**	2.04E+01	6.17E+02	1.79E+01	6.23E+02	1.59E+01	6.54E+02	1.55E+01	6.87E+02	2.85E+01	6.40E+02	1.91E+01
F28	5.25E+02	2.51E+01	**5.23E+02**	2.87E+01	5.28E+02	3.20E+01	5.30E+02	2.49E+01	5.29E+02	1.95E+01	5.25E+02	3.93E+01	5.34E+02	3.34E+01
F29	**8.45E+02**	1.03E+02	8.90E+02	1.50E+02	1.47E+03	1.95E+02	1.44E+03	2.34E+02	1.43E+03	1.72E+02	2.21E+03	2.43E+02	1.23E+03	1.37E+02
F30	2.47E+03	2.37E+02	2.42E+03	1.69E+02	2.53E+03	2.02E+02	2.49E+03	1.99E+02	**2.40E+03**	1.52E+02	3.33E+03	1.43E+03	2.58E+03	1.72E+02

Author contributions Yawei Huang was responsible for methodology and writing—original draft preparation. Yawei Huang, Xuezhong Qian, and Wei Song performed validation. Yawei Huang and Xuezhong Qian contributed to writing—reviewing and editing. Xuezhong Qian and Wei Song were involved in project administration.

Funding This work was supported by the National Natural Science Foundation of China (62076110) and the Natural Science Foundation of Jiangsu Province (BK20181341).

Data availability The data used to support the findings of this study are available from the corresponding author upon request.

Declarations

Conflict of interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Ethical and informed consent for data used All data used in this study are derived from publicly available databases. These data are public and do not involve the use of or conflict of interest of personal privacy; therefore, no specific ethical review and informed consent are required. We have adhered to all appropriate provisions and requirements for using these data.

References

1. Storn R, Price K (1997) Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. J Global Optim 11:341–359
2. Rauf HT, Bangyal WHK, Lali MI (2021) An adaptive hybrid differential evolution algorithm for continuous optimization and classification problems. Neural Comput Appl 33(17):10841–10867
3. Hameed A, Aboobaider B, Mutar M et al (2020) A new hybrid approach based on discrete differential evolution algorithm to enhancement solutions of quadratic assignment problem. Int J Ind Eng Comput 11(1):51–72
4. Li W, Wu B (2019) A modified differential evolution algorithm for constrained optimization problems. In: 2019 2nd World Conference on Mechanical Engineering and Intelligent Manufacturing (WCMEIM). IEEE, pp 69–72
5. Sallam KM, Elsayed SM, Chakrabortty RK, et al (2020) Improved multi-operator differential evolution algorithm for solving unconstrained problems. In: 2020 IEEE Congress on Evolutionary Computation (CEC). IEEE, pp 1–8
6. Zhao F, Zhao L, Wang L et al (2020) An ensemble discrete differential evolution for the distributed blocking flowshop scheduling with minimizing makespan criterion. Expert Syst Appl 160:113678
7. Houssein EH, Rezk H, Fathy A et al (2022) A modified adaptive guided differential evolution algorithm applied to engineering applications. Eng Appl Artif Intell 113:104920
8. Wang R, Fan F, Shen F et al (2021) Application of differential evolution on elasticity measurement of low quality factor materials using fem-based resonant ultrasound spectroscopy. J Mech Behav Biomed Mater 124:104848
9. Kuang B, Xiao C, Wang Z (2021) An enhanced differential evolution for solving extended environmental/economic dispatch. In: 2021 IEEE 5th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC). IEEE, pp 1061–1066
10. Mezura-Montes E, Velázquez-Reyes J, Coello Coello CA (2006) A comparative study of differential evolution variants for global optimization. In: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, pp 485–492
11. Liu G, Xiong C, Guo Z (2015) Enhanced differential evolution using random-based sampling and neighborhood mutation. Soft Comput 19:2173–2192
12. Xia X, Tong L, Zhang Y et al (2021) Nfdde: a novelty-hybrid-fitness driving differential evolution algorithm. Inf Sci 579:33–54

13. Meng Z, Yang C (2022) Two-stage differential evolution with novel parameter control. Inf Sci 596:321–342
14. Price K, Storn RM, Lampinen JA (2006) Differential evolution: a practical approach to global optimization. Springer
15. Zhang J, Sanderson AC (2009) Jade: adaptive differential evolution with optional external archive. IEEE Trans Evol Comput 13(5):945–958
16. Basetti V, Chandel AK, Subramanyam K (2018) Power system static state estimation using jade-adaptive differential evolution technique. Soft Comput 22:7157–7176
17. Li G, Lin Q, Cui L et al (2016) A novel hybrid differential evolution algorithm with modified code and jade. Appl Soft Comput 47:577–599
18. Deng W, Shang S, Cai X et al (2021) Quantum differential evolution with cooperative coevolution framework and hybrid mutation strategy for large scale optimization. Knowl-Based Syst 224:107080
19. Wang Y, Cai Z, Zhang Q (2011) Differential evolution with composite trial vector generation strategies and control parameters. IEEE Trans Evol Comput 15(1):55–66
20. Qian W, Chai J, Xu Z et al (2018) Differential evolution algorithm with multiple mutation strategies based on roulette wheel selection. Appl Intell 48:3612–3629
21. Pant M, Ali M, Abraham A (2009) Mixed mutation strategy embedded differential evolution. In: 2009 IEEE Congress on Evolutionary Computation. IEEE, pp 1240–1246
22. Maučec MS, Brest J, Bošković B et al (2018) Improved differential evolution for large-scale black-box optimization. IEEE Access 6:29516–29531
23. Yang Q, Yan JQ, Gao XD et al (2022) Random neighbor elite guided differential evolution for global numerical optimization. Inf Sci 607:1408–1438
24. Tan Z, Tang Y, Li K et al (2022) Differential evolution with hybrid parameters and mutation strategies based on reinforcement learning. Swarm Evol Comput 75:101194
25. Zhang Y, Dai G, Peng L et al (2023) Enhancing differential evolution algorithm through a population size adaptation strategy. Nat Comput 22(2):379–392
26. Deng L, Li C, Han R et al (2021) Tpde: a tri-population differential evolution based on zonal-constraint stepped division mechanism and multiple adaptive guided mutation strategies. Inf Sci 575:22–40
27. Poláková R, Tvrdík J, Bujok P (2019) Differential evolution with adaptive mechanism of population size according to current population diversity. Swarm Evol Comput 50:100519
28. Li K, Fu X, Wang F et al (2022) A dynamic population reduction differential evolution algorithm combining linear and nonlinear strategy piecewise functions. Concurr Comput Pract Exp 34(6):e6773
29. Zhang X, Zhan ZH, Zhang J (2020) Adaptive population differential evolution with dual control strategy for large-scale global optimization problems. In: 2020 IEEE Congress on Evolutionary Computation (CEC). IEEE, pp 1–7
30. Zeng Z, Zhang M, Chen T et al (2021) A new selection operator for differential evolution algorithm. Knowl-Based Syst 226:107150
31. Zeng Z, Hong Z, Zhang H et al (2022) Improving differential evolution using a best discarded vector selection strategy. Inf Sci 609:353–375
32. Das S, Konar A, Chakraborty UK (2005) Improved differential evolution algorithms for handling noisy optimization problems. In: 2005 IEEE Congress on Evolutionary Computation. IEEE, pp 1691–1698
33. Yu X, Liu Z, Wu X et al (2021) A hybrid differential evolution and simulated annealing algorithm for global optimization. J Intell Fuzzy Syst 41(1):1375–1391
34. Abbas Q, Ahmad J, Jabeen H, et al (2015) A novel tournament selection based differential evolution variant for continuous optimization problems. Math Probl Eng 2015
35. Ghosh A, Das S, Mallipeddi R et al (2017) A modified differential evolution with distance-based selection for continuous optimization in presence of noise. IEEE Access 5:26944–26964
36. Zeng Z, Zhang H (2022) An evolutionary-state-based selection strategy for enhancing differential evolution algorithm. Inf Sci 617:373–394
37. Awad N, Ali M, Liang J, et al (2016) Problem definitions and evaluation criteria for the cec 2017 special session and competition on single objective bound constrained real-parameter numerical optimization. In: Technical Report. Nanyang Technological University Singapore, pp 1–34
38. Biedrzycki R, Arabas J, Warchulski E (2022) A version of nl-shade-rsp algorithm with midpoint for cec 2022 single objective bound constrained problems. In: 2022 IEEE Congress on Evolutionary Computation (CEC). IEEE, pp 1–8

39. Das S, Suganthan PN (2010) Problem definitions and evaluation criteria for cec 2011 competition on testing evolutionary algorithms on real world optimization problems. Jadavpur Univ Nanyang Technol Univ Kolkata 1:341–359
40. Viktorin A, Senkerik R, Pluhacek M et al (2019) Distance based parameter adaptation for success-history based differential evolution. Swarm Evol Comput 50:100462
41. Aggarwal S, Mishra KK (2023) X-mode: extended multi-operator differential evolution algorithm. Math Comput Simul 211:85–108
42. Zhu L, Ma Y, Bai Y (2020) A self-adaptive multi-population differential evolution algorithm. Nat Comput 19:211–235
43. Brest J, Zamuda A, Fister I, et al (2010) Large scale global optimization using self-adaptive differential evolution algorithm. In: IEEE Congress on Evolutionary Computation. IEEE, pp 1–8
44. Qin AK, Huang VL, Suganthan PN (2008) Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Trans Evol Comput 13(2):398–417
45. Meng Z, Song Z, Shao X et al (2023) Fd-de: differential evolution with fitness deviation based adaptation in parameter control. ISA Trans 139:272–290
46. Tanabe R, Fukunaga AS (2014) Improving the search performance of shade using linear population size reduction. In: 2014 IEEE Congress on Evolutionary Computation (CEC). IEEE, pp 1658–1665
47. Tanabe R, Fukunaga A (2013) Success-history based parameter adaptation for differential evolution. In: 2013 IEEE Congress on Evolutionary Computation. IEEE, pp 71–78
48. Zhang X, Liu Q, Qu Y (2023) An adaptive differential evolution algorithm with population size reduction strategy for unconstrained optimization problem. Appl Soft Comput 138:110209
49. Zeng Z, Zhang M, Zhang H et al (2022) Improved differential evolution algorithm based on the sawtooth-linear population size adaptive method. Inf Sci 608:1045–1071
50. Brest J, Maučec MS, Bošković B (2016) il-shade: Improved l-shade algorithm for single objective real-parameter optimization. In: 2016 IEEE Congress on Evolutionary Computation (CEC). IEEE, pp 1188–1195
51. Brest J, Maučec MS, Bošković B (2017) Single objective real-parameter optimization: Algorithm jso. In: 2017 IEEE Congress on Evolutionary Computation (CEC). IEEE, pp 1311–1318
52. Cui L, Huang Q, Li G et al (2018) Differential evolution algorithm with tracking mechanism and backtracking mechanism. IEEE Access 6:44252–44267
53. Guo SM, Yang CC, Hsu PH et al (2014) Improving differential evolution with a successful-parent-selecting framework. IEEE Trans Evol Comput 19(5):717–730
54. Xu C, Huang H, Ye S (2014) A differential evolution with replacement strategy for real-parameter numerical optimization. In: 2014 IEEE Congress on Evolutionary Computation (CEC). IEEE, pp 1617–1624
55. Meng Z, Yang C (2021) Hip-de: historical population based mutation strategy in differential evolution with parameter adaptive mechanism. Inf Sci 562:44–77
56. Meng Z, Zhong Y, Mao G et al (2022) Pso-sono: a novel pso variant for single-objective numerical optimization. Inf Sci 586:176–191
57. Gao Y, Zhang J, Wang Y, et al (2024) Love evolution algorithm: a stimulus–value–role theory-inspired evolutionary algorithm for global optimization. J Supercomput:1–62
58. Ghasemi M, Zare M, Zahedi A et al (2024) Optimization based on performance of lungs in body: lungs performance-based optimization (lpo). Comput Methods Appl Mech Eng 419:116582
59. Mirjalili S, Lewis A (2016) The whale optimization algorithm. Adv Eng Softw 95:51–67