Engineering Applications of Artificial Intelligence 126 (2023) 107001

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

Adaptive differential evolution algorithm based on deeply-informed mutation strategy and restart mechanism

Quanbin Zhang [a], Zhenyu Meng [a,b,*]

[a] Institute of Artificial Intelligence, Fujian University of Technology, Fuzhou, China

[b] Fujian Provincial Key Laboratory of Big Data Mining and Applications, Fujian University of Technology, Fuzhou, China

ARTICLE INFO

Keywords:
Differential evolution
Deeply-informed mutation strategy
Parameters control
Restart mechanism

ABSTRACT

Differential evolution is one of the most powerful stochastic real-parameter optimization algorithms currently, and its performance depends heavily on control parameters and mutation strategy. In recent years, methods to select favorable parameters control and mutation strategy when solving various optimization problems have attracted increasing attention. To choose an appropriate mutation strategy and control parameters for a given optimization problem, in this paper, a Adaptive Differential Evolution Algorithm Based on Deeply-Informed Mutation Strategy and Restart Mechanism (ADEDMR) is proposed, and the ADEDMR algorithm has the following characteristics: First, a deeply-informed mutation strategy is proposed, which takes into account the information of suboptimal solutions discarded by selection and inherits the advantages of the powerful "DE/target-pbest/1/bin", aiming to obtain a better perceptual landscape of the target function and improve the candidate diversity of the trial vector. Second, according to the evolution process, the segmentation method is used to control F, which alleviates the scaling of F in the wrong direction, and makes the newly generated F fit more accurately. Third, a new population restart mechanism is adopted to further enhance population diversity by adaptively enhancing the search ability of hopeless individuals and randomly replacing some inferior individuals with wavelet walks. To evaluate the performance of our proposed algorithm, comparative experiments are conducted on 72 benchmark functions from the CEC2014, CEC2017 and CEC2022 test suites. Experimental results show that the proposed ADEDMR has higher convergence accuracy, better optimization ability when solving high-dimensional complex functions, and is competitive with six recent strong DE variants.

1. Introduction

Optimization plays a key role in the context of maximizing efficiency and reducing the resources required, with all manufacturing and engineering processes being positively impacted. Over past few decades, Global optimization problems have aroused great interest among researchers, to solve these problems, many meta-heuristic algorithms such as particle swarm optimization (PSO) variants (Meng et al., 2022; Bratton and Kennedy, 2007), differential evolution (DE) variants (Storn, 1995; Meng and Chen, 2023; Storn and Price, 1997; Meng, 2023; Das et al., 2016; Meng and Yang, 2021; Song et al., 2023; Li et al., 2023; Liao et al., 2023), ant colony optimization (ACO) variants (Deng et al., 2019; Dorigo et al., 2006), and quasi-affine transformation evolution (QUATRE) variants (Meng and Pan, 2016b,a, 2018; Meng et al., 2020), have been proposed. Owing to their gradient-free search mode (Hu et al., 2015, 2016), they have been applied to solve various real-world applications (Jeong et al., 2010; Meng and Yang, 2022; Li et al., 2020; Zhu et al., 2021; Yu et al., 2023) and various global optimization problems in continuous domains. Since maximization and minimization optimization problems can be transformed into each other, only minimization optimization problems are discussed in this paper. The real-parameter single-objective minimization optimization can be considered as finding a solution X^* from the following set:

$$R^* \equiv \arg\min_{X\in\mathbf{R}} f(X) = \{X^{'} \in \mathbf{R} : f(X^{'}) \leq f(X), \forall X \in \mathbf{R}\} \tag{1}$$

where X represents the D-dimensional vector of the parameters, and $\mathbf{R} \subseteq \mathbb{R}^D$ represents the entire solution space. Generally speaking, the search domain of a vector X is limited by the lower bound $X_{min} = (x_{min,1}, x_{min,2}, \dots, x_{min,D})$ and the upper bound Bound $X_{max} = (x_{max,1}, x_{max,2}, \dots, x_{max,D})$ throughout the iteration. Single-objective numerical optimization problems have various structural and mathematical features. For example, the fitness function used to evaluate the performance of the algorithm can be unimodal, simple multimodal, or a hybrid and composition function (Liang et al., 2013b,a; Wu et al., 2017). Therefore, it is also a hard task to obtain the desired results due to these different characteristics.

* Corresponding author at: Institute of Artificial Intelligence, Fujian University of Technology, Fuzhou, China.
E-mail address: mzy1314@gmail.com (Z. Meng).

https://doi.org/10.1016/j.engappai.2023.107001
Received 27 February 2023; Received in revised form 13 May 2023; Accepted 14 August 2023

Currently, many algorithms have been proposed to solve optimization problems, such as Monte Carlo optimization (Read et al., 2014), swarm intelligence algorithms (Chakraborty and Kar, 2017) and evolutionary algorithms (Vikhar, 2016). However, for numerical optimization problems, evolutionary algorithms have been widely used. Differential Evolution (DE), a classic evolutionary algorithm, was proposed by Storn and Price in 1995 (Storn, 1995; Storn and Price, 1997). DE derived from the genetic annealing algorithm (GAA) (Price et al., 2006), which also combines the advantages of the genetic algorithm (GA) (Holland, 1992) and simulated annealing (SA) (Van Laarhoven and Aarts, 1987). Furthermore, DE also introduces mutation, crossover and selection operations from the above algorithms. Specifically, in the DE algorithm, the mutation strategy determines the search range of each generation, and then the crossover scheme and the selection operation jointly make the choice of the next movement. These three key operations provide a good balance between exploration and exploitation. Fig. 1 shows the search behavior of individuals in DE from a 3-D view, where $X_{i,G}$ is the target vector for individuals in the Gth generation, $V_{i,G}$ represents the mutation vector, and the vertices in the remaining cubes are potential trial vector candidates. From this point of view, the trial vector generation strategy consisting of mutation strategy and crossover scheme dominates the overall performance of DE (Wu et al., 2018). However, it also inherits some disadvantages of the above algorithms, such as the tendency of prematureness, local optimality and low convergence in the later stages of the search. To solve these problems, many scholars have devoted themselves to improving the related work of DE in recent years. Brest et al. (2006) proposed a DE with self-adapting control parameter. Das et al. (2007) proposed an improved DE for automatic clustering of large unlabeled datasets. Zhang and Sanderson proposed JADE to improve optimization performance by implementing the well-known mutation strategy "DE/current-to-pbest" with optional external archive and generating control parameters in an adaptive method. Tanabe and Fukunaga proposed a success-history based adaptive DE (SHADE) (Tanabe and Fukunaga, 2013) and the LSHADE (Tanabe and Fukunaga, 2014) to further extend SHADE, which continuously reduces the population size according to a linear function. Meng et al. (2021) proposed a cooperative strategy to enhance the canonical DE algorithm. Awad et al. (2017) proposed a DE variant of crossover operator with Euclid neighborhood covariance matrix learning. Ali et al. (2016) proposed a multi-population DE and adaptively update mutation and crossover strategies according to the ranking of fitness values. Tian and Gao (2019a) proposed a neighborhood-adaptive differential evolution (NDE), which adjusts the search performance of each individual for better convergence. Brest (Brest et al., 2020) et al. propose a DE with a screening mechanism to select vectors in mutations from subpopulations and employ crowding and restart mechanisms to manage population diversity. Deng et al. (2021) proposed a DE using wavelet basis function to control scale factor F.

Through the review and analysis of these literatures, it can be known that in the past few decades, various mutation operators, multi-swarm and multi-mutation strategies, adaptive control parameter schemes, etc. have been proposed to improve the optimization performance of DE. Although these DE variants achieve good optimization results when solving optimization problems, they still suffer from the defects of falling into local optimum and low convergence. Therefore, in order to solve these defects, an improved DE algorithm named ADEDMR is proposed, whose main contributions are:

1. A deeply-informed mutation strategy is proposed to consider the promising but not-optimal solutions. It can make a better perception of the landscape of objective functions.
2. Adaptive generation of F and CR by segmentation control and Laplace probability distribution, respectively.
3. Enhanced restart mechanism with Wavelet Walk (Salimi, 2015), designed to avoid population stagnation and further accelerate algorithm convergence.

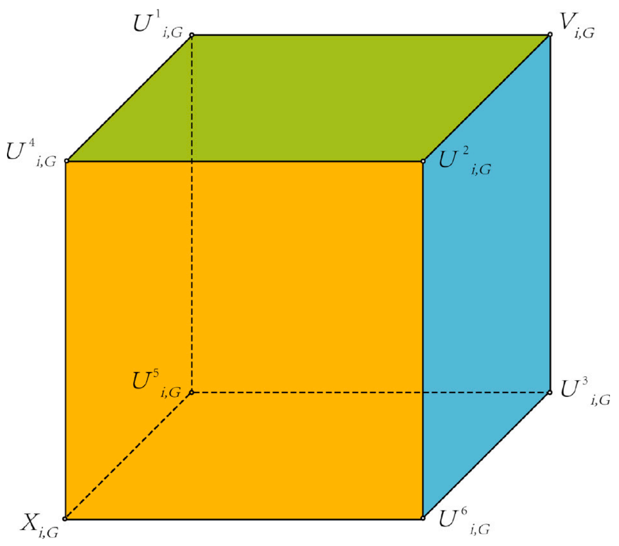


Fig. 1. The hyper-cube search manner in DE variants from a 3-D view.

4. The ADEDMR performance verification uses a total of 60 actual parameter single-objective optimization functions from the CEC2014 (Liang et al., 2013a) and CEC2017 (Wu et al., 2017) test suites, which avoids the over fitting problem to a certain extent compared with the algorithm verification under only one test suite.

The rest of this article is organized as follows. The main flow of the classical DE algorithm is reviewed in Section 2. Section 3 introduces new details of ADEDMR. Section 4 provides numerical results and discussion of ADEDMR. Finally, the conclusions of this paper are presented in Section 5.

2. Classical DE algorithm

Considering the tremendous progress in DE research and its applications in different scientific and technological fields (Das and Suganthan, 2010; Das et al., 2016), this section provides a brief review of classical DE algorithms. The DE algorithm utilizes the differences of individuals in the population to guide the algorithm's search in the solution space. It mainly involves initialization, mutation, crossover, and selection operations, and these three steps may be helpful to effectively understand the improved DE. The main idea of DE is to start from a randomly generated initial population, generate a new individual by summing the vector difference of any two individuals in the population with the third individual, and then select the new individual with the corresponding individual in the current population, through continuous evolution, retain good individuals, eliminate inferior individuals, and guide the search to approach the optimal solution. The basic evolution processes of the DE are described as follows.

2.1. Initialization

The DE algorithm uses a D-dimensional vector (M) as the initial solution. Given the population number (N), each individual can be expressed as $x_i(G) = (x_{i1}(G), x_{i2}(G), \dots, x_{iD}(G))$. The initial population is generated in $[x_{min}, x_{max}]$. Here, M is the number of D-dimensional vectors, N is the number of populations, and $x_i(G)$ is the ith individual.

$$x_{iD} = x_{min} + rand(0,1) \cdot (x_{max} - x_{min}) \tag{2}$$

where G represents generation, x_{max} represents the maximum value of the search space, x_{min} represents the minimum value of the search space, and $rand(0,1)$ represents a random number within $(0,1)$ that satisfies the normal distribution.

2.2. Mutation

The DE algorithm uses mutation operations to generate mutation vectors $V_{i,G}$ for each individual x_i, G in the current population (target vector). Each mutation strategy is denoted by DE/x/y, where x represents the chosen basis vector and y gives the number of difference vectors used. Five widely used mutation strategies (Das et al., 2016) are described below:

(1) **DE/rand/1**

$$V_{i,G} = x_{r_1,G} + F \cdot \left(x_{r_2,G} - x_{r_3,G} \right) \tag{3}$$

(2) **DE/best/1**

$$V_{i,G} = x_{best,G} + F \cdot \left(x_{r_1,G} - x_{r_2,G} \right) \tag{4}$$

(3) **DE/rand − to − best/1**

$$\begin{aligned} V_{i,G} = x_{i,G} + F \cdot (x_{\text{best },G} - x_{i,G}) \\ +F \cdot (x_{r_1,G} - x_{r_2,G}) \end{aligned} \tag{5}$$

(4) **DE/best/2**

$$\begin{aligned} V_{i,G} = x_{best,G} + F \cdot \left(x_{r_1,G} - x_{r_2,G} \right) \\ +F \cdot (x_{r_3,G} - x_{r_4,G}) \end{aligned} \tag{6}$$

(5) **DE/rand/2**

$$V_{i,G} = x_{i,G} + F \cdot (x_{r_1,G} - x_{i,G}) + F \cdot (x_{r_2,G} - x_{r_3,G}) \tag{7}$$

In the above mutation rules, r_1, r_2, r_3, r_4 and r_5 are randomly generated exclusive integers within [1, M]. The scale factor F is a control parameter to amplify difference vector. $x_{best,G}$ is best individual vector in the population at Gth generation.

2.3. Crossover

In the DE algorithm, the binomial crossover operator selects the trial vectors $u_{i,G}(u_{1,G}, u_{2,G}, \dots, u_{i,G})$ between $v_{i,G}$ and $x_{i,G}$. According to the crossover strategy, the trial vector is generated by the following formula:

$$u_{i,G} = \begin{cases} v_{i,G} & if \quad rand_j(0, 1 \leq CR) or j = j_{rand}, j = 1, 2, \dots, D \\ x_{i,G} & otherwise \end{cases} \tag{8}$$

where $rand_j(0, 1)$ is a uniformly distributed random number from the interval(0,1). CR is a crossover rate that determines how much information the generated trial vector u_i inherits from the mutation vector v_i or the target vector x_i.

2.4. Selection

The fitness value of all test vectors are evaluated, and the selection operation is performed. For the minimization optimization problem, if the fitness value of the test individual is lower than or equal to that of the corresponding target individual, the target individual is replaced by test individual and then the population is updated, otherwise the target individual will remain in the population for the next generation. This greedy selection operation can be expressed as follows:

$$X_{i,G+1} = \begin{cases} U_{i,G} & if \quad f\left(U_{i,G}\right) \leq f\left(X_{i,G}\right) \\ X_{i,G} & otherwise \end{cases} \tag{9}$$

3. Proposed ADEDMR algorithm

The classical DE in solving various types of optimization problems requires appropriate selection of mutation strategies and control parameters, which shows the dependence of its search performance on the appropriate choice of mutation strategy and control parameter settings (Mohamed et al., 2021). Mutation strategies addressed in the literature use individuals of DE populations in different ways to implement multiple search patterns in search loops. During the entire search process, most of these strategies either use only random individuals of the population to determine promising search directions, or use a few elite individuals to guide the search direction, while ignoring the guiding role of promising but non-optimal solutions. In this section, the above problem can be well solved by using ADEDMR with a deeply-informed mutation strategy, controlling the parameters according to the evolutionary state of the individuals participating in the mutation strategy and updating the individuals.

3.1. Motivation

It is well known that the performance of DE depends heavily on the mutation strategy and parameter control. Specifically, the mutation operation determines the range of the space to be explored or exploited, while the control parameters adjust the search ability. However, different problems and stages of evolution often require different mutation operators and parameter settings. Therefore, it is greatly meaningful to design some mutation strategies and parameter settings with adaptive adjustment ability. Moreover, the depth information between individuals and different mutation operators are usually integrated to generate offspring, and the historical search information of the current population is often used to control parameters (Zhang and Sanderson, 2009; Tanabe and Fukunaga, 2013; Li et al., 2014; Tian and Gao, 2019a; Meng and Pan, 2019). However, how to use effective information to make the evolutionary direction as promising as possible is a difficult problem. Therefore, we use the promising but non-optimal solutions eliminated after the selection operation to integrate with the mutation operator of the current population to achieve a better fit. This idea comes from the marathon. The whole process of the competition is relatively long, and the players who are relatively behind during the competition may eventually win a good ranking.

For parameter control, many studies in the past have shown that the performance of adaptive parameter control is often better than the fixed value method. The generation of the scaling factor F of many powerful DE variants follows the Cauchy distribution, but this method cannot retain the previous some good F, and may even develop in a worse direction. The generation of F obeys the Cauchy distribution and successful historical archive in alternate generation, aiming at reducing the false development of F. After subsequent polishing, the segmentation method in this paper is finally obtained. The core idea behind ADEDMR is described as follows.

3.2. Deeply-informed mutation strategy

As the DE's core operator, mutation has received extensive attention since the birth of DE. The mutation strategy can be simply expressed as a weighted expression between the basis vector and the difference vector. The basic vector is used to guide and regulate the evolution direction of the population, and the difference vector plays the role of random disturbance and fine search. Furthermore, each of the five commonly used variational strategies has its own characteristics. DE/rand/1 strategy can handle single-peak and multi-peak optimization problems better, but has poor convergence. DE/rand/2 strategy has better global search capability, but slower convergence. DE/best/1 and DE/best/2 strategies have faster convergence, but their global exploration capability is relatively weak and they tend to fall into local convergence. DE/rand-to-best/1 strategy has relatively balanced global exploration and local optimization, but relatively poor robustness. For different complex optimization problems, each variant strategy has different optimization capabilities. Therefore, some DE variants use candidate pools with mutation strategies. However, it essentially trade off space for time, and the time complexity is high.

Various recent DE variants suggest that embedding elite individuals to guide the search direction may be an effective way to improve DE performance. However, most algorithms use only one elite individual to guide the others during evolution, which can lead to premature convergence. Furthermore, this way of guiding does not always provide a promising search direction from low to high fitting, which can lead to hopeless search behaviors, some of which may be misled into invalid search regions of the solution space. In the deeply informed mutation strategy we proposed, in addition to using the top 100%p elite individuals with fitness values, the depth information integrated with the suboptimal solution and the current population is also used. Specifically, the information is extracted from the promising individuals eliminated by the selection operation, and then integrated with the current population to generate a difference vector. The purpose is to adjust the evolutionary direction of the population and avoid falling into local optimum. The proposed deeply-informed mutation strategy formula is as follows:

$$V_{i,G} = X_{i,G} + F \cdot (X^{p}_{best,G} - X_{i,G}) + F \cdot (\widetilde{X}_{r_1,G} - \widehat{X}_{r_2,G}) \tag{10}$$

where $X_{i,G}$ denotes the individual vector in the current population. Similar to JADE (Zhang and Sanderson, 2009), $X^{p}_{best,G}$ represents top 100%p the vector randomly selected from the current population. The early stage of evolution requires more exploration of the global region, while the later stage of evolution exploits the local region more, and the corresponding settings of the p-values change dynamically with the evolutionary process, $p \in 0.2 \sim 0.05$. The indices of i, r_1 and r_2 are always not equal to each other ($i \neq r_1 \neq r_2$). $\widetilde{X}_{r_1}$ is a vector containing promising but not optimal individuals integrated with X_{r_1}, where P and X_{r_1} denote the population of current individuals and randomly selected vector from P, respectively. In order to search in the direction of high fitting as much as possible, individuals whose fitness value is greater than or equal to the average fitness value of the current population are removed from the selection process, while the retained individuals are stored in the archive A. The size of the archive changes adaptively with the current population size ps, $|A| = r^{arc}_{A} \cdot ps_G$, where r^{arc}_{A} represents the externally archived constant ratio population size, the recommended value is 0.6. $\widehat{X}_{r_2,G}$ denotes a vector randomly selected from the union $P \cup B$, B denotes the external archived solution set with a time stamp scheme. B is similar to the time stamp external archive in PaDE, but the difference is that we use a new time stamp mechanism to better adapt to problems of different scales. The time stamp mechanism in PaDE needs to set a fixed threshold and a decay rate of r^d. We have introduced a dynamic threshold in the archive based on the ranking of fitness values in order to maintain a better balance between the number of general solutions and the number of poor solutions. In the ADEDMR algorithm, the calculation of r^d is as follows:

$$r^d = \frac{R_G(i)}{R_G(1)} \tag{11}$$

where $R_G(\cdot)$ represents the number of individuals whose fitness value of the current population is better than that of the previous generation. In addition, the inferior solutions of the current population are also added to the external archive at the end of each generation, if the size of the external archive exceeds a fixed maximum value, $r^{arc}_{B} \cdot ps$, or when $r^d = 0$, then randomly pick and eliminate many solutions from the archive, keeping the archive size equal to $r^{arc}_{B} \cdot ps$.

3.3. Parameter adaption

$$F = \mu_F \tag{12}$$

$$F_i = \begin{cases} Cauchy(\mu_F, 0.1) & if\ FS < 0.1 \\ \mu_F & if\ FS >= 0.1 \\ Cauchy(\mu_F, 0.1) & while\ F_i \leq 0 \\ 1 & if\ F_i > 1 \\ F_i & otherwise \end{cases} \tag{13}$$

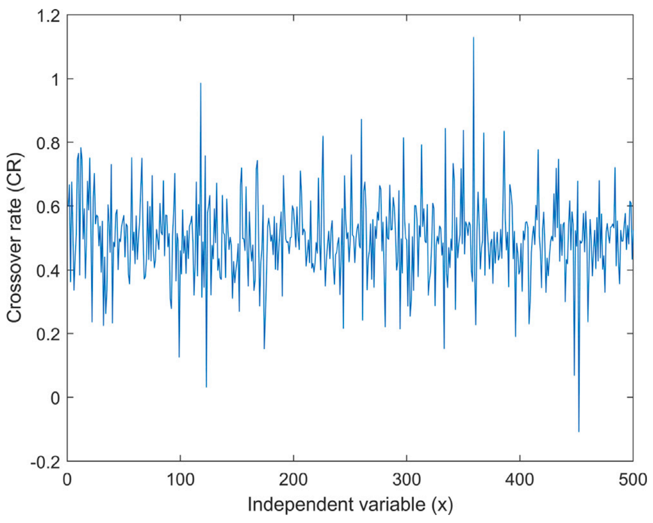


Fig. 2. The value of the crossover rate CR.

where the Cauchy($\cdot$) represents the random number generated by the Cauchy distribution. μ_F and 0.1 of Cauchy($\cdot$) represent the location parameter and scale parameter, respectively., FS represents scale factor status, and its calculation formula is as follows:

$$FS = \frac{F - min(F)}{max(F) - min(F)} \tag{14}$$

where $min(F)$ and $max(F)$ represent the minimum and maximum values of F. If there is a situation where $min(F)$ is equal to $max(F)$, skip the calculation of FS and enter the next step.

$$CR_i = Laplace(\mu_{CR}, 0.1) \tag{15}$$

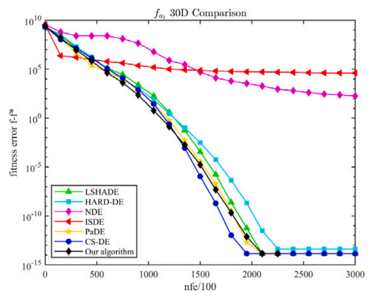
$$CR_i = \begin{cases} 0 & if\ \mu_{CR} \leq 0 \parallel CR_i < 0 \\ 1 & if\ \mu_{CR} > 0 \&\& CR_i > 1 \\ CR_i & otherwise \end{cases} \tag{16}$$

where μ_F and μ_{CR} represent history archives of F and CR, their initial value is 0.5. The variance of CR under the Laplace distribution is set to 0.1. The probability density of the Laplace distribution looks similar to the normal distribution, but it is more concentrated than the normal distribution. While having the properties of the normal distribution, it can produce more extreme values, which means that it produces a larger range disturbance. To intuitively understand the Laplace distribution, the values of CR are shown in Fig. 2.

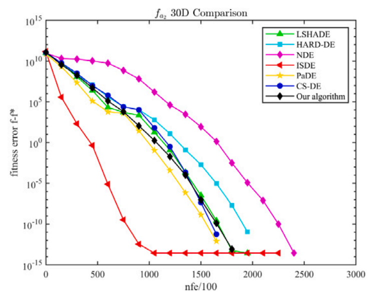
3.4. Restart mechanism

Through some investigations (Neri and Tirronen, 2010; Das and Suganthan, 2010; Yang et al., 2014; Das et al., 2016), it is found that two key factors affecting DE performance are premature convergence and stagnation. Currently, several mechanisms have been developed to alleviate the evolutionary dilemmas (Tian and Gao, 2019a; Tian et al., 2020). However, these mechanisms either only adjust for the search ability of hopeless individuals, or do not take population information into account to measure evolutionary status. In order to solve the above problems and speed up the convergence, a restart mechanism considering the effect of population diversity is developed to further enhance the performance of the algorithm.

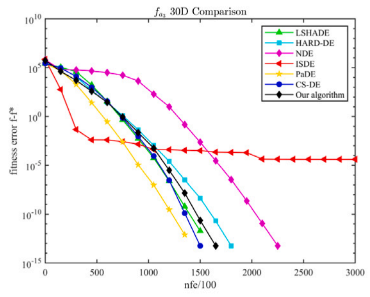
Here, the diversity measure technique used considers the computation of two hypervolumes (Osuna-Enciso et al., 2022): one is the constraint of the search space, and the other represents the spatial distribution of the population in the iterations. The space limit volume



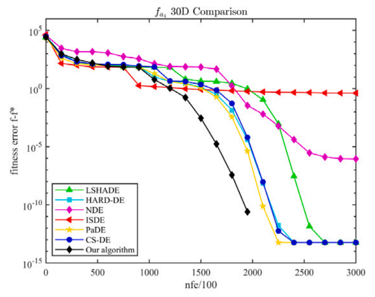
(a) Benchmark f_{a_1}



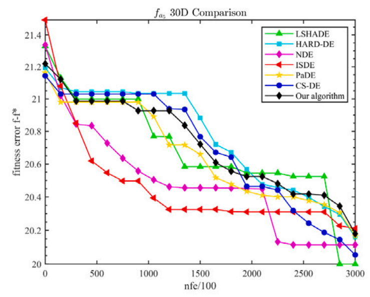
(b) Benchmark f_{a_2}



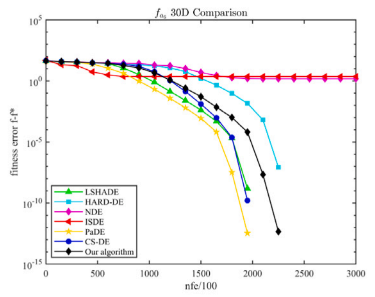
(c) Benchmark f_{a_3}



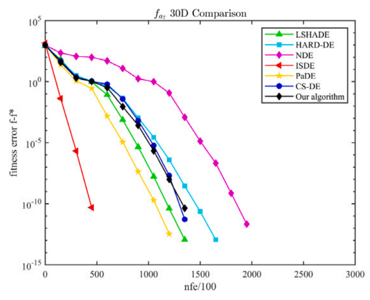
(d) Benchmark f_{a_4}



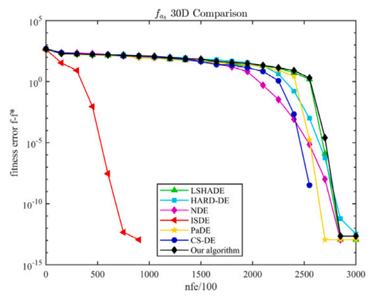
(e) Benchmark f_{a_5}



(f) Benchmark f_{a_6}



(g) Benchmark f_{a_7}



(h) Benchmark f_{a_8}

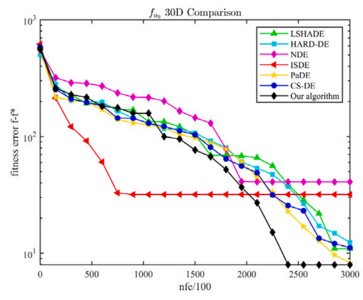
Fig. 3. Here presents the convergence speed comparison by employing the median value of 51 runs obtained by each algorithm on 30-D optimization under benchmarks f_{a_1}–$f_{a_{30}}$ of our test suite. There are total 30 comparison figures and the first 8 figures are presented here.

is calculated by searching for the absolute value of the difference between the upper and lower bounds of the space.

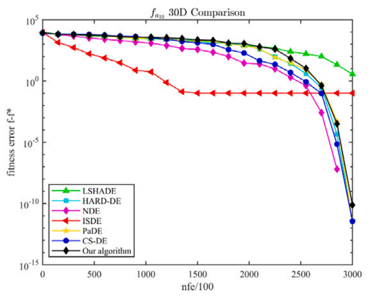
$$V_{lim} = \ln(1 + \prod_{i=1}^{D} |u_i - l_i|) \tag{17}$$

$$V_{pop} = \sqrt{\prod_{i=1}^{D} \mathbf{y}_i} \tag{18}$$

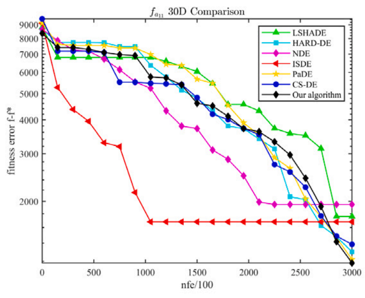
where $l = l_1, l_2 \cdots, l_D$ and $u = u_1, u_2 \cdots, u_D$ are vectors which represents the lower and upper limits of the search space, respectively. And V_{lim} is computed only once during the initialization phase. The second hypervolume V_{pop} denotes the evolving population. $\mathbf{y}_i$ means the edge vector coordinates: $\mathbf{y}_i = (2 \cdot xnd(\mathbf{x}_i), 0, 0, \ldots)$. $xnd(\mathbf{x}_i)$ is the result of the maximum minus the minimum of the column vector for each individual in the current generation. Furthermore, the ratio between the population of candidate solutions and the hypervolume restricted by the search space is adopted as the diversity measure. It is computed



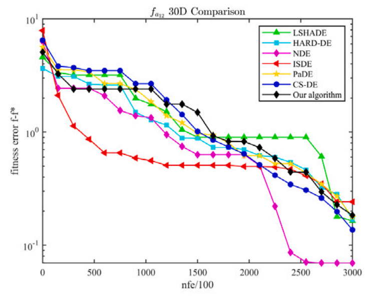
(a) Benchmark f_{a_9}



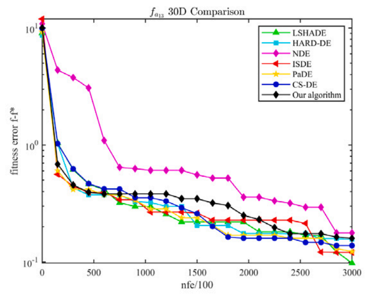
(b) Benchmark $f_{a_{10}}$



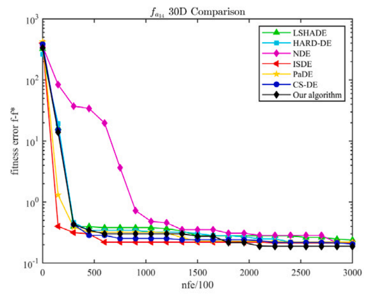
(c) Benchmark $f_{a_{11}}$



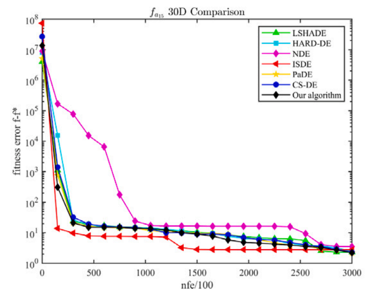
(d) Benchmark $f_{a_{12}}$



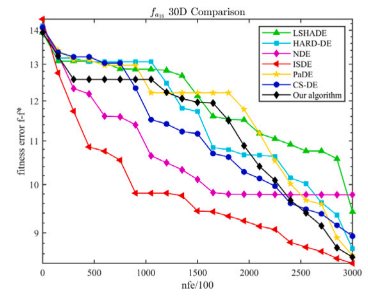
(e) Benchmark $f_{a_{13}}$



(f) Benchmark $f_{a_{14}}$



(g) Benchmark $f_{a_{15}}$



(h) Benchmark $f_{a_{16}}$

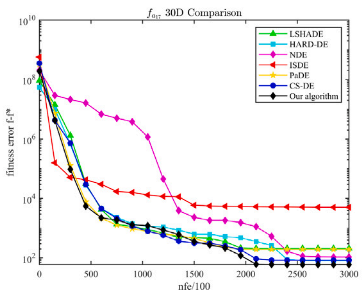
Fig. 4. As a continued part from Fig. 3, convergence comparisons on benchmarks $f_{a_9} - f_{a_{16}}$ are given here.

in Eq. (19).

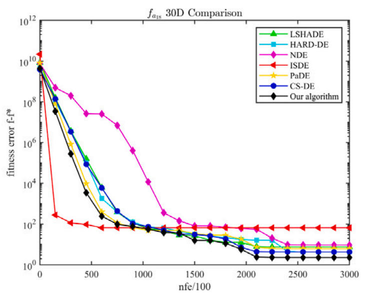
$$i_{VOL} = \sqrt{\frac{V_{pop}}{V_{lim}}} \tag{19}$$

It is named i_{VOL} because it utilizes the multidimensional volume of the individual search space as the basis for its process. Moreover, the method is simple and uses the ratio between the D-dimensional volume bounded by the search space and the similar hypervolume representing the overall population as a diversity metric. For the sake of clarification, the metrics are described in the Algorithm 1.

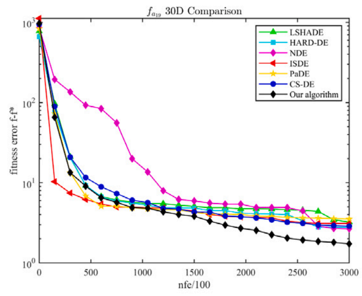
Furthermore, we set up a counter ct to record consecutive generations of individuals without progress. When the population does not converge to a smaller area, the i_{VOL} is bigger than ξ and the counter ct is bigger than $N(i_{VOL} > \xi, ct > N)$, $N = 40$ and ξ=0.001. The next



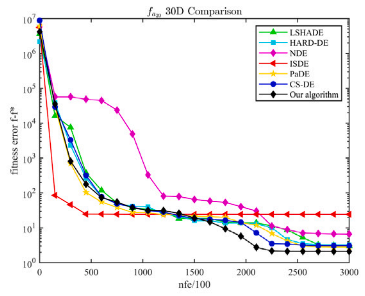
(a) Benchmark $f_{a_{17}}$



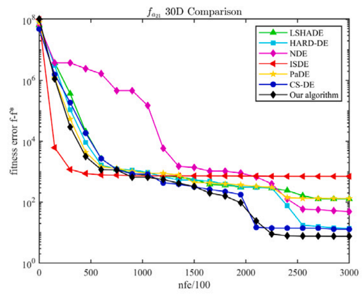
(b) Benchmark $f_{a_{18}}$



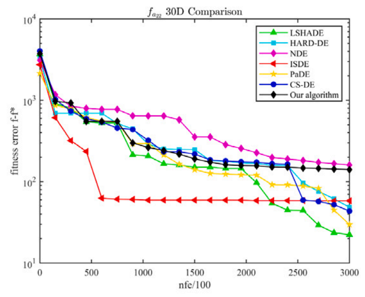
(c) Benchmark $f_{a_{19}}$



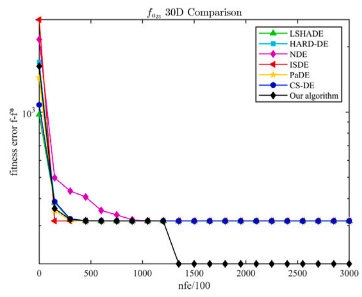
(d) Benchmark $f_{a_{20}}$



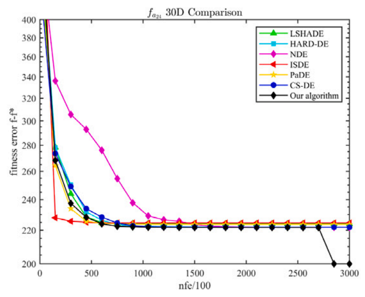
(e) Benchmark $f_{a_{21}}$



(f) Benchmark $f_{a_{22}}$



(g) Benchmark $f_{a_{23}}$



(h) Benchmark $f_{a_{24}}$

Fig. 5. As a continued part from Fig. 4, convergence comparisons on benchmarks $f_{a_{17}} - f_{a_{24}}$ are given here.

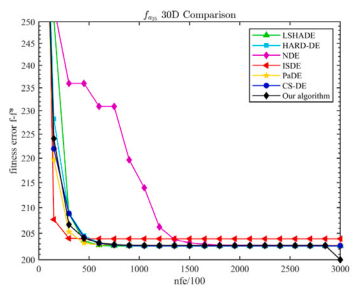
step is to perform a population convergence operation:

$$X_{i,G}^{new} = X_{i,G} + F_{restart} \cdot (X_{r_1,G}^{better} - X_{i,G}) \tag{20}$$

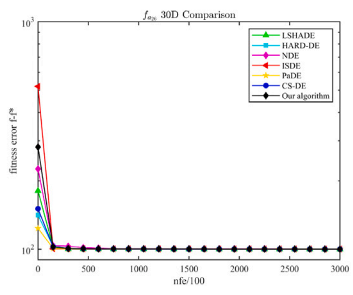
where $X_{r_1,G}^{better}$ is an individual randomly selected from individuals ranked higher than $X_{i,G}$ in the population. If the rank of the ith individual in the entire population is k, then the individual ranked in $[1, k-1]$ can be selected as $X_{r_1,G}^{better}$. $X_{i,G}^{new}$ denotes the newly generated individual, and then performs a selection operation with the current individual $X_{i,G}$, and the remaining outstanding individuals replace the current individual $X_{i,G}$. The calculation of scale factor $F_{restart}$ uses a Cauchy distribution model. The calculation is as follows:

$$F_{restart} = Cauchy(0.5, 0.1) \tag{21}$$

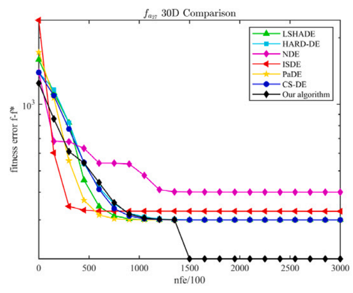
When the diversity of the population is insufficient ($i_{VOL} < \xi$), unpromising individuals in the population are replaced by more promising individuals. Therefore, using Wavelet Walk can effectively improve the search performance of unpromising individuals, avoiding population stagnation and trapping in local optima. Specifically, an elite small population X_e is selected from the current population for Wavelet Walk, and some inferior individuals in population are replaced by a



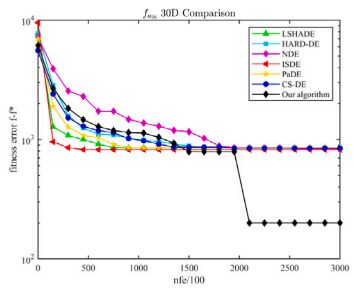
(a) Benchmark f_{a25}



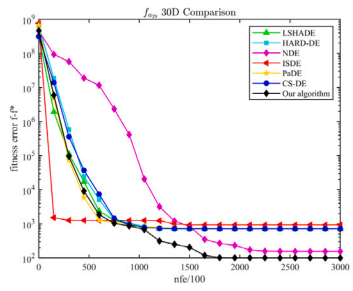
(b) Benchmark f_{a26}



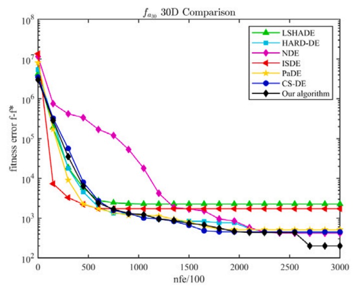
(c) Benchmark f_{a27}



(d) Benchmark f_{a28}



(e) Benchmark f_{a29}



(f) Benchmark f_{a30}

Fig. 6. As a continued part from Fig. 5, the last four comparisons are given here.

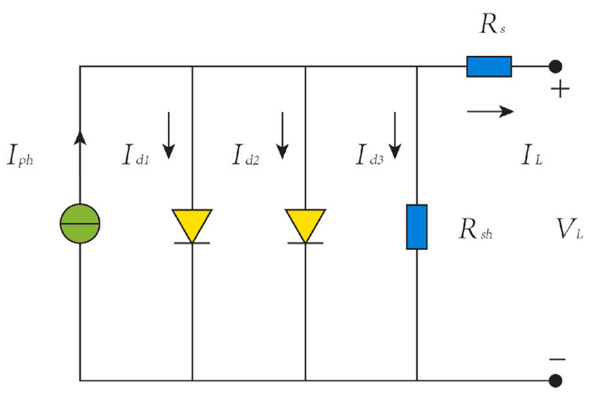


Fig. 7. Schematics for double diode model.

more promising individual generated by Wavelet Walk. For the sake of clarification, the proposed Wavelet Walk is explained in Algorithm 2.

As can be seen from the Algorithm 2, X_{gbest} denotes global optimal fitness value. X_e is a small population containing top 10 individuals in the current population. EP denotes the number of X_e, which is set to 10 in this paper. $fitness_{X_e}$ and $fitness_X$ represent small elite population X_e and population fitness value, respectively. rnd_{index} means randomly

Algorithm 1 Pseudocode for the calculation of i_{VOL}

```
Data: population of candidate solutions $X_i$
  $V_{pop} = 1$
  for j = 1 to D do
      $V_{pop} = \sqrt{V_{pop} \cdot |max(X(:,j)) - min(X(:,j))|}$
  end for
  $i_{VOL} = \sqrt{\frac{V_{pop}}{V_{lim}}}$
  return Individual Diversity Metric ($i_{VOL}$).
```

picking an integer from $[1, ps]$. Furthermore, the wavelet function we use is called the Mexican hat wavelet function, which is also the second derivative of the Gaussian function, so it inherits some properties of the Gaussian distribution and has good localization properties in both the time and frequency domains.

3.5. Parameter control

In ADEDMR, a memory pool is set up containing H entries, each with an argument pair μ_F and μ_{CR}. The control parameters F and CR are generated by randomly selecting a parameter pair from one entry in the memory pool at a time. After greedy selection, if a better test vector is obtained, the corresponding individual is regarded as a

Algorithm 2 Procedure of Wavelet Walk

```
Input: $X_e$, $EP$, $G$, $X_{gbest}$, $X$, $fitness_{X_e}$, $fitness_X$;
Output: $fitness_X$, $X_e$;
  Select elite individuals $X_e$ and global optimal individuals $X_{gbest}$ from the current population
  if $i_{VOL} < 0.001$ then
    for $i = 1{:}EP$ do
      if $G = 1$ then
        $X_{e,i} = X_{e,i} + \frac{e}{\sqrt{5}} \cdot \pi^{-\frac{1}{4}} \cdot (X_{e,i} - X_{gbest})^2 \cdot e^{-\frac{(X_{e,i}-X_{gbest})^2}{2}}$
      else
        $X_{e,i} = \frac{e}{\sqrt{5}} \cdot \pi^{-\frac{1}{4}} \cdot (X_{e,i} - X_{gbest})^2 \cdot e^{-\frac{(X_{e,i}-X_{gbest})^2}{2}}$
      end if
      Boundary restrictions and fitness evaluation
      if then $fitness_{X_e}(i) < fitness_X(rnd_{index}(i))$
        $fitness_X(rnd_{index}(i)) = fitness_{X_e}(i)$
        $X_i = X_{e,i}$
        $ct(rnd_{index}(i)) = 0$
      end if
    end for
  end if
```

successful individual, denoted as "s", otherwise it is regarded as a failed individual, denoted as "f". Before updating the parameters μ_F and μ_{CR}, we need to count the number of times each entry in the memory pool is selected and calculate its success rate R. To avoid the situation where the success rate is 0, a small constant ε (ε=0.01) is set. The calculation formula of the success rate R is as follows:

$$R_j = \begin{cases} \frac{n_{s,j}^2}{n_s \cdot (ns_j + nf_j)} & if\ \ n_{s,j} > 0 \\ \varepsilon & otherwise \end{cases} \tag{22}$$

where $j \in [1, 2, \ldots, H]$, n_s is the number of individuals marked as s in current population, and $n_s = n_{s,1} + n_{s,2} + \cdots + n_{s,H-1} + n_{s,H}$. $n_{s,j}$ is the number of individuals marked as s between the individuals who selected the jth entry, $n_j = n_{s,j} + n_{f,j}$. In ADEDMR, the update method of μ_{CR} in the memory pool is the same as that in LSHADE, all from the first entry to the last entry, and then repeat this cycle. For parameter μ_F, only the μ_F value of the entry corresponding to the smallest success rate R is replaced each time. This can be equivalent to using the principle of control variables to mitigate mis-interaction between parameters. The adaptation scheme of μ_F and μ_{CR} were calculated according to Eqs. (23) and (24).

$$\begin{cases} w_s = \frac{std(\Delta loc_i)}{\sum_{s=1}^{|S_F|} std(\Delta loc_i)} \\ \Delta loc_i = loc(U_{i,G} - X_{i,G}) \\ mean_{WL}(S_F) = \frac{\sum_{s=1}^{|S_F|} w_s \cdot S_F^2(s)}{\sum_{s=1}^{|S_F|} w_s \cdot S_F(s)} \\ \mu_{F,idx,G+1} = \begin{cases} mean_{WL}(S_F), & if\ \ S_F \neq \emptyset \\ \mu_{F,idx,G}, & otherwise \end{cases} \end{cases} \tag{23}$$

$$\begin{cases} w_s = \frac{std(\Delta loc_i)}{\sum_{s=1}^{|S_F|} std(\Delta loc_i)} \\ \Delta loc_i = loc(U_{i,G} - X_{i,G}) \\ mean_{WL}(S_{CR}) = \frac{\sum_{s=1}^{|S_{CR}|} w_s \cdot S_{CR}^2(s)}{\sum_{s=1}^{|S_{CR}|} w_s \cdot S_{CR}(s)} \\ \mu_{CR,k,G+1} = \begin{cases} mean_{WL}(S_{CR}), & if\ \ S_{CR} \neq \emptyset \& \max\{CR\} > 0 \\ 0, & if\ \ S_{CR} \neq \emptyset \max\{CR\} = 0 \\ \mu_{CR,k,G}, & otherwise \end{cases} \end{cases} \tag{24}$$

where S_F and S_{CR} represent the set of F and CR values that yield better trial vectors, respectively. k represents the index in the memory pool, which repeats cyclically from the first entry to the last. $loc(U_{i,G} - X_{i,G})$ represents the position difference between the test vector $U_{i,G}$ and the target vector $X_{i,G}$, $std(\cdot)$ means to calculate the standard deviation. Using the position information of the population instead of the fitness value information to participate in the adaptation of the control parameters makes the DE algorithm have a wider application, especially for some objective functions where the fitness value is infeasible. In addition, the linear population size reduction scheme proposed in LSHADE proved to be a good population size adaptation scheme, but the rapid reduction of population size at the beginning of evolution often leads to some objective function perception errors of the landscape. For this problem, a better parabolic model is proposed in HARD-DE. On this basis, we use a better parabolic model to further optimize the accuracy of the curve, which is also applicable to other algorithms. The detailed equation is as follows:

$$ps_{G+1} = \begin{cases} ceil[\frac{ps_{\min} - ps_{ini}}{(\frac{2}{3}nfe_{\max} - ps_{ini})^2} \cdot (nfe - ps_{ini})^2 + ps_{ini}], \\ \qquad if \quad nfe \le 0.5nfe_{\max} \\ floor[\frac{ps_{\min} - \frac{1}{3}ps_{ini}}{\frac{1}{3}nfe_{\max}} \cdot (nfe - nfe_{\max}) + ps_{\min}], \\ \qquad otherwise \end{cases} \tag{25}$$

4. Numerical results and discussion

To evaluate the performance of ADEDMR, this section analyzes the performance of ADEDMR from the aspects of experimental numerical analysis, convergence curve, mutation strategy, parameter setting and time complexity. Since the analysis process is the same for all benchmark problem sets, considering the layout factor, the experimental numerical analysis is only carried out on CEC2014, CEC2017 and CEC2022. Using the 72 benchmarks under both test suites also avoids overfitting problems that can occur when algorithms use smaller test suites.

4.1. Experimental environment and numerical analysis

All the experimental environment performed on Enterprise 64-bit version of MATLAB 2021a on a personal computer equipped with an Intel(R) Core(TM) i7-11700K@3.6 GHz CPU and Microsoft Windows 10 system. The algorithm was run 51 times on each benchmark, and the statistics, mean, and standard deviation of the fitness error $f - f^*$ were calculated from the results of the 51 runs for experimental analysis. To have statistically sound conclusions, Wilcoxon rank sum test (Wilcoxon, 1992) is employed to show the differences between two algorithms on a single problem, with the significant level $\alpha = 0.05$. For clarity, the initial settings of parameters for different DE variants are shown in Table 1.

To evaluate the benefits of ADEDMR, we compare ADEDMR with 6 well-known optimization algorithms from a total of 72 benchmarks in CEC2014, CEC2017 and CEC2022 in Table 2- Table 9. These algorithms include LSAHDE (Tanabe and Fukunaga, 2014), HARD-DE (Meng and Pan, 2019), NDE (Tian et al., 2020), ISDE (Tian and Gao, 2019b), PaDE (Meng et al., 2019), CS-DE (Meng et al., 2021). The symbols " > ", " ≈ " and " < " in parentheses after the value indicate "better performance", "similar performance" and "worse performance" of ADEDMR. Furthermore, the bottom of each table also summarizes the performance of each algorithm on the test suite, and the performance comparison in Table 2- Table 9 are summarize in Table 12. The results show that ADEDMR is very competitive with these six powerful DE variants under the CEC2014, CEC2017 and CEC2022 test sets optimizations. For the optimization under the CEC2014 test suite, it can be concluded from Table 12 that ADEDMR is highly competitive with the six DE variants compared (including LSHADE, HADR-DE, NDE, ISDE, PaDE and CS-DE) in our test suite of 72 test suites. Different from the summarized results in Table 11, Table 2- Table 4 present the

Table 1
Recommended parameter settings of all these contrasted algorithms.

Algorithm	Parameters initial settings
LSHADE	$\mu_F = 0.5,\ F \sim C(\mu_F, 0.1),\ \mu_{CR} = 0.5,\ CR \sim C(\mu_C R, 0.1),\ ps = 18 \cdot D \sim 4,\ r^{rac} = 2.6,\ p = 0.11,\ H = 6$
HARD-DE	$\mu_F = 0.3,\ \mu_C R = 0.8,\ F\&CR$ same as LSHADE, $p = 0.11,\ ps = 25 \cdot ln(D) \cdot \sqrt{D} \sim 4,\ r^{rac,A} = 1.6,\ r^{rac,B} = 3,\ k = 4$
NDE	$\mu_F = \mu_{CR} = 0.5,\ ps = 10 \cdot D \sim 5,\ gm = 10,\ c = 0.1$
ISDE	$ps = 50,\ K = 100,\ \alpha = 0.6,\ \beta = \gamma = 0.5,\ freq = 0.01$
PaDE	$\mu_F = 0.8,\ \mu_C R = 0.6,\ F\&CR$ same as LSHADE, $p = 0.11,\ ps = 25 \cdot ln(D) \cdot \sqrt{D} \sim 4,\ r^{rac,A} = 1.6,\ T_0 = 70,\ r^d = 0.04$
CS-DE	$\mu_F = 0.6,\ \mu_{CR} = 0.8,\ F\&CR$ same as LSHADE, $p = 0.25 \sim 0.05,\ K = 6,\ r^{rac,A} = 1.6,\ r^{rac,B} = 5,\ NP = 25 \cdot ln(D) \cdot \sqrt{D} \sim K,\ T_0 = \frac{G_{\max}}{2}$
ADEMDR	$\mu_F = \mu_{CR} = 0.5,\ F \sim L(\mu_F, 0.2),\ CR \sim L(\mu_C R, 0.1),\ ps = 18 \cdot D \sim 10,\ n = 40,\ \xi = 0.001,\ H = 5,\ p = 0.2 \sim 0.05,\ r^{rac,A} = 0.6,\ r^{rac,B} = 1.6$

Table 2
Comparisons results of **Mean** and **St**andard **d**eviation (Mean/Std) of fitness errors $f - f^*$ over 51-run between several well-known DE variants and ADEDMR under f_{a_1} - $f_{a_{30}}$ of CEC2014 test suite on 10D optimization. The fixed maximum number of function evaluations nfe_{max} equaling to 10000 ·D. The overall performance (>, =or<) of each algorithm is measured under Wilcoxon's signed rank test with the significant level $\alpha = 0.05$ in comparison with the proposed ADEDMR. The bottom line summarized the overall performance on the 30 benchmarks.

DE Variants:	LSHADE	HARD-DE	NDE	ISDE	PaDE	CS-DE	ADEDMR
f_{a_1}	0.0000E+00/0.0000E+00(≈)	0.0000E+00/0.0000E+00(≈)	0.0000E+00/0.0000E+00(≈)	4.2633E−14/1.4746E−13(<)	0.0000E+00/0.0000E+00(≈)	0.0000E+00/0.0000E+00(≈)	0.0000E+00/0.0000E+00
f_{a_2}	0.0000E+00/0.0000E+00(≈)	0.0000E+00/0.0000E+00(≈)	0.0000E+00/0.0000E+00(≈)	0.0000E+00/0.0000E+00(≈)	0.0000E+00/0.0000E+00(≈)	0.0000E+00/0.0000E+00(≈)	0.0000E+00/0.0000E+00
f_{a_3}	0.0000E+00/0.0000E+00(≈)	0.0000E+00/0.0000E+00(≈)	0.0000E+00/0.0000E+00(≈)	1.2333E−04/8.8077E−04(≈)	0.0000E+00/0.0000E+00(≈)	0.0000E+00/0.0000E+00(≈)	0.0000E+00/0.0000E+00
f_{a_4}	3.0006E+01/1.2088E+01(<)	1.3297E+01/1.6758E+01(>)	9.6306E+00/1.4943E+01(>)	2.7022E+01/1.4165E+01(≈)	2.5318E+01/1.5547E+01(≈)	2.4551E+01/1.6005E+01(≈)	2.2502E+01/1.6784E+01
f_{a_5}	1.4500E+01/8.6229E+00(≈)	**9.6468E+00/9.7166E+00**(>)	1.8856E+01/4.7612E+00(<)	1.6978E+01/7.2638E+00(<)	1.2161E+01/8.9643E+00(≈)	1.2680E+01/9.2188E+00(≈)	1.4132E+01/9.2129E+00
f_{a_6}	0.0000E+00/0.0000E+00(≈)	1.2665E−02/3.9334E−02(<)	1.7556E−02/1.2526E−01(<)	1.7863E−02/1.2524E−01(≈)	0.0000E+00/0.0000E+00(≈)	0.0000E+00/0.0000E+00(≈)	0.0000E+00/0.0000E+00
f_{a_7}	1.2076E−03/3.7636E−03(>)	7.6846E−07/4.1929E−06(>)	8.1546E−02/5.2748E−02(<)	2.2495E−02/1.7650E−02(<)	3.5748E−03/6.3340E−03(≈)	4.3511E−04/1.7575E−03(>)	3.9119E−03/8.0765E−03
f_{a_8}	0.0000E+00/0.0000E+00(≈)	1.5604E−14/3.9511E−14(<)	1.9509E−02/1.3932E−01(≈)	0.0000E+00/0.0000E+00(≈)	0.0000E+00/0.0000E+00(≈)	0.0000E+00/0.0000E+00(≈)	0.0000E+00/0.0000E+00
f_{a_9}	2.2261E+00/8.3573E−01(<)	2.6603E+00/1.0064E+00(<)	7.9987E+00/3.3650E+00(<)	3.5116E+00/1.4938E+00(<)	2.3258E+00/7.3560E−01(<)	2.3432E+00/8.8559E−01(<)	**1.7931E+00/1.2253E+00**
$f_{a_{10}}$	2.4492E−03/1.2244E−02(>)	5.7066E−13/4.4410E−13(>)	3.1840E−02/5.7746E−02(≈)	1.9393E−01/6.6638E−01(<)	4.8984E−03/1.6958E−02(>)	**3.5666E−14/1.7830E−13**(>)	3.0615E−02/3.8240E−02
$f_{a_{11}}$	3.5873E+01/3.9130E+01(≈)	3.2653E+01/3.7811E+01(≈)	1.2361E+02/1.2893E+02(<)	7.0028E+01/8.6359E+01(<)	2.6300E+01/2.8542E+01(≈)	5.1564E+01/5.4261E+01(<)	4.0865E+01/8.4252E+01
$f_{a_{12}}$	6.8063E−02/1.6525E−02(≈)	7.1881E−02/4.7431E−02(≈)	**3.7284E−02/5.4658E−02**(>)	2.3611E−01/9.2348E−02(<)	9.0799E−02/3.2253E−02(<)	5.6307E−02/1.9091E−02(≈)	6.4001E−02/3.2045E−02
$f_{a_{13}}$	4.8490E−02/1.4094E−02(≈)	5.4454E−02/2.3762E−02(≈)	8.7847E−02/3.9160E−02(<)	9.5115E−02/4.6772E−02(<)	4.8641E−02/1.4137E−02(≈)	4.6245E−02/1.4420E−02(≈)	5.2020E−02/1.6516E−02
$f_{a_{14}}$	**6.7410E−02/2.9017E−02**(>)	7.8068E−02/2.7547E−02(>)	1.4123E−01/4.6344E−02(<)	9.0935E−02/3.0186E−02(≈)	8.5169E−02/2.5778E−02(≈)	8.3786E−02/2.7375E−02(≈)	9.1202E−02/3.6244E−02
$f_{a_{15}}$	3.6887E−01/6.9947E−02(≈)	4.1118E−01/1.0380E−01(<)	8.5617E−01/3.4723E−01(<)	5.2826E−01/1.7450E−01(<)	3.9808E−01/6.7612E−02(≈)	3.9159E−01/8.4557E−02(≈)	3.7115E−01/7.2697E−02
$f_{a_{16}}$	1.2052E+00/3.6554E−01(<)	1.3099E+00/2.6316E−01(<)	1.6154E+00/6.5686E−01(<)	9.9824E−01/4.3514E−01(≈)	9.9508E−01/3.2954E−01(≈)	1.2500E+00/2.9064E−01(<)	9.1768E−01/3.7221E−01
$f_{a_{17}}$	8.0677E−01/8.9995E−01(>)	2.8082E+00/2.7405E+00(<)	5.7351E+00/1.0152E+01(≈)	1.6901E+01/2.9746E+01(<)	6.7040E−01/7.4298E−01(>)	1.5751E+00/1.3749E+00(≈)	1.4838E+00/1.9092E+00
$f_{a_{18}}$	2.2721E−01/1.9491E−01(<)	1.1353E−01/1.2403E−01(<)	1.2599E+00/1.0885E+00(<)	9.0538E−01/6.4159E−01(<)	1.1140E−01/1.1490E−01(<)	8.9012E−02/8.7888E−02(<)	**4.9698E−02/5.1362E−02**
$f_{a_{19}}$	7.5432E−02/4.3044E−02(<)	7.0911E−02/4.0886E−02(<)	2.5951E−01/2.5452E−01(<)	5.3647E−02/4.3100E−02(<)	5.3516E−02/2.6529E−02(<)	5.1289E−02/2.5036E−02(<)	**3.9460E−02/3.3344E−02**
$f_{a_{20}}$	1.5630E−01/1.1892E−01(>)	1.7789E−01/7.7607E−02(≈)	5.5116E−01/5.5487E−01(<)	**7.6569E−02/1.5698E−01**(>)	1.1694E−01/1.2425E−01(>)	2.0841E−01/1.5181E−01(≈)	2.4379E−01/1.9360E−01
$f_{a_{21}}$	3.4841E−01/2.2690E−01(≈)	4.6975E−01/3.3708E−01(≈)	1.3780E+00/5.1366E+00(≈)	1.0976E+00/3.2093E+00(<)	2.5773E−01/2.2981E−01(≈)	2.8076E−01/2.3166E−01(≈)	3.2607E−01/2.5924E−01
$f_{a_{22}}$	9.2008E−02/5.3493E−02(>)	1.0999E−01/4.3848E−02(≈)	1.9827E−01/2.9351E−01(≈)	9.6769E−02/1.5087E−01(≈)	8.2190E−02/3.2127E−02(>)	8.9365E−02/3.3358E−02(>)	1.2703E−01/4.8132E−02
$f_{a_{23}}$	3.2946E+02/0.0000E+00(<)	3.2946E+02/0.0000E+00(<)	3.2946E+02/0.0000E+00(<)	3.2946E+02/4.5927E−13(<)	3.2946E+02/0.0000E+00(<)	3.2946E+02/0.0000E+00(<)	**2.0000E+02/0.0000E+00**
$f_{a_{24}}$	1.0776E+02/1.7309E+00(<)	1.0699E+02/2.7789E+00(≈)	1.1502E+02/4.9097E+00(<)	1.0985E+02/3.2030E+00(<)	1.0649E+02/2.2496E+00(≈)	1.0785E+02/2.1342E+00(<)	1.0605E+02/2.9750E+00
$f_{a_{25}}$	1.4329E+02/4.3743E+01(<)	1.2800E+02/3.0557E+01(<)	1.2107E+02/6.1782E+00(<)	1.6211E+02/4.1663E+01(<)	1.2274E+02/2.6140E+01(≈)	1.1890E+02/5.4279E+00(<)	**1.1978E+02/2.4756E+01**
$f_{a_{26}}$	1.0005E+02/1.6938E−02(≈)	1.0006E+02/2.9864E−02(≈)	1.0009E+02/4.9212E−02(<)	1.0010E+02/4.8056E−02(<)	1.0006E+02/1.7953E−02(≈)	1.0005E+02/1.5928E−02(≈)	1.0005E+02/1.7856E−02
$f_{a_{27}}$	6.9742E+01/1.3283E+02(≈)	1.3195E+01/5.8521E+01(≈)	2.0626E+00/6.6451E−01(<)	1.9674E+02/1.5432E+02(<)	4.0325E+01/1.0930E+02(≈)	1.3283E+01/5.8503E+01(≈)	1.3161E+01/4.7176E+01
$f_{a_{28}}$	3.7841E+02/3.0807E+01(<)	3.8840E+02/4.9293E+01(<)	3.6400E+02/2.4425E+01(<)	3.7308E+02/2.2718E+01(<)	3.9002E+02/5.1787E+01(<)	3.7740E+02/4.1861E+01(<)	**2.6888E+02/9.1401E+01**
$f_{a_{29}}$	2.2212E+02/5.5963E−01(<)	2.2179E+02/1.9277E−01(<)	**1.7188E+02/5.2489E+01**(>)	2.2353E+02/1.3560E+00(<)	2.2285E+02/2.8845E−01(<)	2.2181E+02/2.2877E−01(<)	2.0043E+02/3.0481E+00
$f_{a_{30}}$	4.6549E+02/1.0424E+01(<)	4.7051E+02/1.4520E+01(<)	4.8293E+02/2.7992E+01(<)	4.8194E+02/2.4752E+01(<)	4.6994E+02/9.2280E+00(<)	4.6332E+02/3.6363E+00(<)	**2.6595E+02/1.1034E+02**
>/≈/<	6/13/11	5/12/13	3/8/19	1/8/21	4/18/8	3/16/11	–/–/–

detailed results of 10D, 30D and 50D real-parameter single-objective optimization under the CEC2014 test suite, respectively. As can be seen from Table 2, ADEDMR can find the global optimum on f_{a_1}, f_{a_2}, f_{a_3}, f_{a_6} and f_{a_8}, and it also secures the best or similar performance on $f_{a_1} - f_{a_3}$, f_{a_6}, f_{a_8}, f_{a_9}, $f_{a_{11}}$, $f_{a_{13}}$, $f_{a_{15}}$, $f_{a_{16}}$, $f_{a_{18}}$, $f_{a_{19}}$, $f_{a_{21}}$, $f_{a_{23}} - f_{a_{28}}$ and $f_{a_{30}}$ in comparison with the other six DE variants. From Table 3 we can see that ADEDMR can find the global optimum on f_{a_2}, f_{a_3}, f_{a_6} and f_{a_7}, and it also ensures the best or tier best performance on $f_{a_1} - f_{a_4}$, f_{a_6}, f_{a_7}, f_{a_9}, $f_{a_{11}}$, $f_{a_{14}}$, $f_{a_{17}} - f_{a_{21}}$, $f_{a_{23}} - f_{a_{30}}$ in comparison with the other six DE variants. From Table 4 we can assures the best or tier best performance on $f_{a_1} - f_{a_4}$, f_{a_6}, f_{a_7}, $f_{a_{11}}$, $f_{a_{14}} - f_{a_{21}}$, $f_{a_{23}} - f_{a_{25}}$, $f_{a_{27}} - f_{a_{30}}$ in comparison with the other six DE variants. For the convenience of observation, we summarize the algorithm performance comparison under 30D in Tables 13 and 14. It is noteworthy that ADEDMR achieves the best or results on almost all hybrid functions ($f_{a_{17}}$–$f_{a_{22}}$) and composition functions ($f_{a_{23}}$–$f_{a_{30}}$) under the CEC2014 test benchmark.

For the optimization under the CEC2017 test suite, the results in Table 5- Table 7 also support the superiority of ADEDMR compared to the other six DE variants. Moreover, ADEDMR can show excellent or similar performance on all 30 benchmark functions compared to CS-DE at 10D and 50D, ISDE at 30D and 50D, and NDE at 50D.

For the optimization under the CEC2022 test suite, the results in Tables 8 and 9 also support the superiority of ADEDMR compared to the other six DE variants. Moreover, ADEDMR exhibits 3 better and 9 similar performances compared to PADE in 10D, and 6 better and 5 similar performances compared to ISDE in 20D. Interestingly, when comparing the optimization results, the superiority of ADEDMR increases gradually with the increase of dimensionality. The main reason for this may be our parameter adaptive control.

4.2. Convergence curve analysis

To further demonstrate the convergence performance, we have done an experiment to compare the convergence speed optimization of each algorithm under the CEC2014 30D test suite. Convergence curves reflect the median of 51 runs of each algorithm obtained on the 30-D optimization and divide the number of iterations into 21 equal parts. Figs. 3–6 depict the evolution curves of ADEDMR and six DE variants under CEC2014 30D. From these convergence curves we can see that ADEDMR obtains better or similar performance in comparison with LSHADE on benchmarks f_{a_2}, $f_{a_9} - f_{a_{11}}$, $f_{a_{14}} - f_{a_{21}}$, $f_{a_{23}} - f_{a_{25}}$, $f_{a_{27}} - f_{a_{30}}$; outperforms HARD-DE on benchmarks $f_{a_1} - f_{a_4}$, $f_{a_6} - f_{a_9}$, $f_{a_{11}}$, $f_{a_{14}} - f_{a_{21}}$, $f_{a_{23}} - f_{a_{25}}$, $f_{a_{27}} - f_{a_{30}}$; outperforms NDE on benchmarks $f_{a_1} - f_{a_4}$, f_{a_6}, f_{a_7}, f_{a_9}, $f_{a_{11}}$, $f_{a_{13}} - f_{a_{25}}$, $f_{a_{27}} - f_{a_{30}}$; outperforms ISDE on benchmarks $f_{a_1} - f_{a_6}$, $f_{a_9} - f_{a_{12}}$, $f_{a_{14}}$, $f_{a_{15}}$, $f_{a_{17}} - f_{a_{30}}$; outperforms PaDE on benchmarks f_{a_1}, f_{a_4}, f_{a_9}, $f_{a_{11}}$, $f_{a_{14}} - f_{a_{21}}$, $f_{a_{23}} - f_{a_{25}}$, $f_{a_{27}} - f_{a_{30}}$; outperforms CS-DE on benchmarks on f_{a_1}, f_{a_4}, f_{a_9}, $f_{a_{11}}$, $f_{a_{14}} - f_{a_{21}}$, $f_{a_{23}} - f_{a_{25}}$, $f_{a_{27}} - f_{a_{30}}$. Furthermore, We can also see that our ADEDMR algorithm can even find global optima on some benchmarks, such as f_{a_2}, f_{a_3}. f_{a_6} and f_{a_7}. As a result, the proposed ADEDMR algorithm is competitive with these powerful DE variants from convergence speed view, especially on hybrid and composition functions Table 6.

Table 3
Comparisons results of **Mean** and **St**andard **d**eviation (Mean/Std) of fitness errors $f - f^*$ over 51-run between several well-known DE variants and ADEDMR under f_{a_1} - $f_{a_{30}}$ of CEC2014 test suite on 30D optimization. The fixed maximum number of function evaluations nfe_{max} equaling to 10000 ·D. The overall performance (>, =or<) of each algorithm is measured under Wilcoxon's signed rank test with the significant level $\alpha = 0.05$ in comparison with the proposed ADEDMR. The bottom line summarized the overall performance on the 30 benchmarks.

DE Variants:	LSHADE	HARD-DE	NDE	ISDE	PaDE	CS-DE	ADEDMR
f_{a_1}	1.2818E−14/5.1277E−15(<)	5.2385E−14/4.2868E−14(<)	8.0984E+02/1.7321E+03(<)	6.1542E+04/4.8321E+04(<)	1.3096E−14/6.2548E−15(<)	7.5234E−15/7.1637E−15(<)	**2.7864E−15/5.6983E−15**
f_{a_2}	*0.0000E+00/0.0000E+00(≈)*	*2.2292E−15/7.7172E−15(≈)*	*0.0000E+00/0.0000E+00(≈)*	0.0000E+00/7.0884E−14(<)	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00*
f_{a_3}	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00(≈)*	4.9144E−02/2.2657E−01(<)	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00*
f_{a_4}	1.1252E−10/8.0302E−10(<)	6.1302E−14/2.5019E−14(<)	8.5269E−04/3.1794E−03(<)	1.6408E+00/8.8221E+00(<)	3.9010E−14/2.6638E−14(<)	4.2354E−14/2.7481E−14(<)	**1.1146E−15/7.9597E−15**
f_{a_5}	**2.0017E+01/7.6771E−02**(>)	2.0166E+01/8.4135E−02(≈)	2.0122E+01/8.2122E−02(>)	2.0225E+01/9.0621E−02(<)	2.0172E+01/5.4905E−02(≈)	2.0057E+01/1.9322E−02(>)	2.0183E+01/8.5040E−02
f_{a_6}	*9.0057E−03/6.4312E−02(≈)*	2.4028E−05/7.4239E−05(<)	1.4684E+00/1.5831E+00(<)	2.6359E+00/1.9429E+00(<)	*0.0000E+00/0.0000E+00(≈)*	*2.1348E−04/1.1227E−03(≈)*	*0.0000E+00/0.0000E+00*
f_{a_7}	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/1.6078E−14(≈)*	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00*
f_{a_8}	1.4935E−13/5.3276E−14(>)	3.3883E−13/7.3643E−14(<)	2.3411E−01/1.2833E+00(>)	**0.0000E+00/0.0000E+00**(>)	1.2929E−13/7.5478E−14(>)	4.0125E−14/5.4870E−14(>)	3.5221E−13/5.2604E−13
f_{a_9}	1.0301E+01/2.8893E+00(<)	1.2677E+01/2.3163E+00(<)	4.4767E+01/1.6893E+01(<)	3.0903E+01/7.8962E+00(<)	*8.1116E+00/1.7219E+00(≈)*	1.1290E+01/1.5231E+00(<)	*8.8634E+00/4.2692E+00*
$f_{a_{10}}$	4.3104E+00/3.2603E+00(<)	3.1743E−12/9.5167E−13(>)	**2.4771E−02/1.7690E−01**(>)	3.4310E−01/4.6495E−01(<)	1.2247E−03/4.9474E−03(>)	4.0822E−04/2.9153E−03(>)	2.4493E−03/7.9518E−03
$f_{a_{11}}$	1.7771E+03/3.5881E+02(<)	1.2807E+03/1.8158E+02(≈)	2.0120E+03/5.0729E+02(<)	1.6023E+03/3.5557E+02(<)	*1.2054E+03/1.8417E+02(≈)*	1.3605E+03/2.2963E+02(<)	*1.2188E+03/3.2984E+02*
$f_{a_{12}}$	1.8117E−01/9.5812E−02(≈)	1.7520E−01/3.3889E−02(≈)	**8.4256E−02/5.7323E−02**(>)	2.3701E−01/8.2248E−02(<)	1.7743E−01/3.5897E−02(≈)	1.3733E−01/2.0793E−02(>)	1.8080E−01/4.6190E−02
$f_{a_{13}}$	**1.0142E−01/2.9136E−02**(>)	1.4914E−01/2.4224E−02(≈)	1.7925E−01/4.9013E−02(<)	1.3558E−01/4.8200E−02(>)	1.1738E−01/1.6572E−02(>)	1.3982E−01/1.6167E−02(>)	1.5589E−01/2.4722E−02
$f_{a_{14}}$	2.3420E−01/3.8124E−02(<)	2.0878E−01/3.2660E−02(<)	2.1369E−01/4.3337E−02(<)	2.0596E−01/3.4106E−02(<)	2.1430E−01/2.3504E−02(<)	2.0208E−01/2.3705E−02(<)	**1.8729E−01/2.0456E−02**
$f_{a_{15}}$	2.3570E+00/4.8618E−01(≈)	2.4004E+00/2.6156E−01(<)	3.5458E+00/1.2260E+00(<)	2.8896E+00/6.5287E−01(<)	*2.1696E+00/2.0954E−01(≈)*	2.2984E+00/1.9508E−01(≈)	*2.2360E+00/3.6399E−01*
$f_{a_{16}}$	9.3223E+00/6.9720E−01(<)	8.7435E+00/4.5803E−01(<)	9.8330E+00/8.9880E−01(<)	*8.3622E+00/6.2258E−01(≈)*	*8.5587E+00/4.3723E−01(≈)*	8.9307E+00/3.6120E−01(<)	*8.4712E+00/5.1129E−01*
$f_{a_{17}}$	2.1528E+02/1.0769E+02(<)	9.7937E+01/4.9006E+01(<)	1.9983E+02/1.8898E+02(<)	6.6672E+03/5.2047E+03(<)	2.1631E+02/8.6515E+01(<)	1.1885E+02/7.0091E+01(<)	**5.9657E+01/1.4654E+01**
$f_{a_{18}}$	7.8433E+00/3.1996E+00(<)	4.3610E+00/1.4790E+00(<)	9.8318E+00/4.1142E+00(<)	1.2377E+02/2.0724E+02(<)	5.5529E+00/2.8097E+00(<)	4.5593E+00/2.2742E+00(<)	**2.3046E+00/1.3040E+00**
$f_{a_{19}}$	3.2191E+00/5.9149E−01(<)	2.8845E+00/5.1382E−01(<)	2.6477E+00/5.2034E−01(<)	3.2166E+00/7.7805E−01(<)	3.5460E+00/4.6598E−01(<)	3.0874E+00/5.0684E−01(<)	**1.7758E+00/5.5869E−01**
$f_{a_{20}}$	2.9142E+00/1.3130E+00(<)	3.1501E+00/1.1583E+00(<)	6.8915E+00/2.4395E+00(<)	2.9612E+01/1.9626E+01(<)	2.8883E+00/1.2583E+00(<)	2.9940E+00/1.1243E+00(<)	**2.1475E+00/9.4406E−01**
$f_{a_{21}}$	1.0964E+02/8.2097E+01(<)	4.2258E+01/5.1602E+01(<)	1.1305E+02/1.1351E+02(<)	1.0414E+03/1.0438E+03(<)	8.9820E+01/8.4078E+01(<)	4.1418E+01/6.0206E+01(<)	**1.0928E+01/2.1014E+01**
$f_{a_{22}}$	**2.4344E+01/3.9285E+00**(>)	8.2432E+01/5.5492E+01(≈)	1.7816E+02/8.4292E+01(<)	1.0572E+02/9.2479E+01(≈)	5.4074E+01/4.7178E+01(>)	7.5614E+01/5.4552E+01(>)	9.8007E+01/5.7007E+01
$f_{a_{23}}$	3.1524E+02/0.0000E+00(<)	3.1524E+02/0.0000E+00(<)	3.1524E+02/2.1434E−10(<)	3.1524E+02/9.1854E−13(<)	3.1524E+02/3.7327E−13(<)	3.1524E+02/0.0000E+00(<)	**2.0000E+02/0.0000E+00**
$f_{a_{24}}$	2.2426E+02/1.8070E+00(<)	2.2185E+02/3.2262E+00(<)	2.1957E+02/7.1665E+00(<)	2.2503E+02/1.7834E+00(<)	2.2356E+02/9.4045E−01(<)	2.2226E+02/8.0396E−01(<)	**2.0085E+02/4.2578E+00**
$f_{a_{25}}$	2.0260E+02/5.2188E−02(<)	2.0261E+02/5.5169E−02(<)	2.0277E+02/1.9560E−01(<)	2.0438E+02/1.4154E+00(<)	2.0272E+02/1.5229E−01(<)	2.0261E+02/4.6865E−02(<)	**2.0036E+02/8.9944E−01**
$f_{a_{26}}$	*1.0010E+02/3.1952E−02*(>)	1.0014E+02/2.5208E−02(>)	1.0018E+02/4.2996E−02(<)	1.0014E+02/5.6465E−02(≈)	*1.0011E+02/1.6501E−02*(>)	1.0014E+02/1.7303E−02(>)	1.0015E+02/2.1741E−02
$f_{a_{27}}$	3.0000E+02/1.2531E−13(<)	3.0196E+02/1.4003E+01(<)	4.0043E+02/2.4786E−01(<)	3.4551E+02/4.0900E+01(<)	3.0000E+02/1.1139E−13(<)	3.0000E+02/4.9447E−05(<)	**2.0000E+02/0.0000E+00**
$f_{a_{28}}$	8.5170E+02/1.1170E+01(<)	8.5031E+02/1.6611E+01(<)	8.3767E+02/2.9918E+01(<)	8.2049E+02/2.8124E+01(<)	8.6042E+02/1.5856E+01(<)	8.4686E+02/1.7622E+01(<)	**2.0000E+02/0.0000E+00**
$f_{a_{29}}$	7.1672E+02/3.7165E+00(<)	5.9753E+02/2.2929E+02(<)	1.5131E+02/2.7928E+01(<)	1.0297E+03/2.6016E+02(<)	6.9699E+02/1.0568E+02(<)	7.0670E+02/7.5372E+01(<)	**1.2501E+02/2.8523E+01**
$f_{a_{30}}$	2.2634E+03/8.1675E+02(<)	4.5779E+02/7.6096E+01(<)	4.1611E+02/3.1373E+01(<)	1.7969E+03/6.4252E+02(<)	5.7611E+02/1.7654E+02(<)	4.6134E+02/6.0427E+01(<)	**2.0000E+02/0.0000E+00**
>/≈/<	5/6/19	2/8/20	4/3/23	2/4/24	5/10/15	7/5/18	–/–/–

Table 4
Comparisons results of **Mean** and **St**andard **d**eviation (Mean/Std) of fitness errors $f - f^*$ over 51-run between several well-known DE variants and ADEDMR under f_{a_1} - $f_{a_{30}}$ of CEC2014 test suite on 50D optimization. The fixed maximum number of function evaluations nfe_{max} equaling to 10000 ·D. The overall performance (>, =or<) of each algorithm is measured under Wilcoxon's signed rank test with the significant level $\alpha = 0.05$ in comparison with the proposed ADEDMR. The bottom line summarized the overall performance on the 30 benchmarks.

DE Variants:	LSHADE	HARD-DE	NDE	ISDE	PaDE	CS-DE	ADEDMR
f_{a_1}	6.2586E+02/1.2222E+03(<)	2.8629E+01/8.0323E+01(<)	1.4271E+05/8.1568E+04(<)	5.6353E+05/2.5159E+05(<)	8.2822E+02/9.5278E+02(<)	7.3601E+01/9.7110E+01(<)	**1.3644E−09/2.7526E−09**
f_{a_2}	3.5666E−14/1.2510E−14(<)	6.5203E−14/2.3639E−14(<)	1.2474E−02/2.8631E−02(<)	4.0208E+03/4.0287E+03(<)	4.3468E−14/1.4327E−14(<)	*2.8979E−14/3.9798E−15(≈)*	*2.8422E−14/0.0000E+00*
f_{a_3}	5.4614E−14/1.1144E−14(<)	5.6843E−14/0.0000E+00(<)	5.2676E−04/1.6709E−03(<)	9.0230E+02/1.8550E+03(<)	5.5729E−14/7.9597E−15(<)	*4.2354E−14/2.5019E−14(≈)*	*4.4583E−14/2.3612E−14*
f_{a_4}	5.0118E+01/4.7598E+01(<)	1.1082E+01/2.9604E+01(<)	2.1003E+01/2.3032E+01(<)	7.9925E+01/2.2662E+01(<)	7.8919E+00/2.6580E+01(<)	6.2761E+00/2.3190E+01(<)	**1.9452E+00/1.3734E+01**
f_{a_5}	**2.0041E+01/1.4039E−01**(>)	2.0266E+01/7.9727E−02(>)	2.0216E+01/1.2071E−01(>)	2.0269E+01/1.0499E−01(>)	2.0267E+01/8.5312E−02(>)	2.0141E+01/2.3834E−02(>)	2.0342E+01/9.0558E−02
f_{a_6}	2.6793E−01/5.7124E−01(<)	1.4772E−02/1.0500E−01(<)	8.2735E+00/6.8994E+00(<)	1.0299E+01/3.8673E+00(<)	1.7953E−01/4.3979E−01(<)	6.2765E−04/8.5002E−04(<)	**5.5377E−07/3.9547E−06**
f_{a_7}	8.0250E−14/5.2316E−14(<)	8.4708E−14/5.0038E−14(<)	*0.0000E+00/0.0000E+00(≈)*	4.3506E−04/1.7576E−03(<)	8.0250E−14/5.2316E−14(<)	*4.4583E−15/2.2287E−14(≈)*	*8.9166E−15/3.0869E−14*
f_{a_8}	3.3864E−08/2.0426E−07(>)	6.5314E−13/1.4352E−13(>)	1.1315E+00/3.1778E+00(≈)	*5.8527E−02/2.3644E−01*(>)	7.1110E−13/1.7293E−13(>)	*2.0954E−13/2.1615E−13*(>)	7.0251E−08/9.3304E−08
f_{a_9}	*1.5002E+01/5.0335E+00*(>)	2.5426E+01/2.9110E+00(<)	7.2277E+01/3.7762E+01(<)	7.5305E+01/1.5955E+01(<)	*1.5602E+01/2.4899E+00*(>)	2.0963E+01/2.7391E+00(≈)	2.2192E+01/1.1619E+01
$f_{a_{10}}$	3.9125E+01/3.0085E+01(<)	*4.4110E−03/8.2181E−03*(>)	1.6963E−01/5.2329E−01(<)	5.5176E+00/2.3302E+01(<)	*6.9959E−03/9.6597E−03*(>)	*2.9441E−03/6.4118E−03*(>)	1.0748E−02/1.1566E−02
$f_{a_{11}}$	4.2457E+03/6.2098E+02(<)	3.3588E+03/2.4410E+02(≈)	4.5546E+03/8.1931E+02(<)	3.7866E+03/8.0443E+02(<)	*3.1802E+03/3.1513E+02(≈)*	*3.2740E+03/2.9271E+02(≈)*	*3.2757E+03/4.0320E+02*
$f_{a_{12}}$	3.3583E−01/1.5380E−01(<)	2.2800E−01/4.3430E−02(≈)	**1.1394E−01/8.1671E−02**(>)	2.2959E−01/1.1370E−01(≈)	2.2484E−01/3.8149E−02(>)	1.7456E−01/2.2574E−02(>)	2.4387E−01/6.0673E−02
$f_{a_{13}}$	**1.4192E−01/3.0508E−02**(>)	2.1533E−01/3.4661E−02(>)	2.3980E−01/5.6705E−02(≈)	2.3495E−01/4.3836E−02(≈)	1.8761E−01/2.0691E−02(>)	1.9881E−01/2.0534E−02(>)	2.3594E−01/2.8194E−02
$f_{a_{14}}$	3.2871E−01/3.2358E−02(<)	2.7429E−01/2.5000E−02(<)	2.5458E−01/3.5969E−02(<)	3.0014E−01/9.1970E−02(<)	2.8985E−01/2.2126E−02(<)	2.8013E−01/2.3988E−02(<)	**2.3029E−01/1.7568E−02**
$f_{a_{15}}$	*4.9285E+00/6.9861E−01(≈)*	5.3211E+00/4.3870E−01(<)	5.7909E+00/1.8303E+00(<)	5.6686E+00/1.1174E+00(<)	*5.0886E+00/4.6623E−01(≈)*	5.2380E+00/4.7233E−01(≈)	*5.0047E+00/7.4545E−01*
$f_{a_{16}}$	1.8085E+01/7.8263E−01(<)	1.7011E+01/3.8201E−01(<)	1.8620E+01/9.9162E−01(<)	*1.6220E+01/1.0067E+00(≈)*	1.6727E+01/5.6795E−01(≈)	1.7156E+01/4.2580E−01(<)	*1.6605E+01/4.8650E−01*
$f_{a_{17}}$	1.4910E+03/3.8387E+02(<)	7.1199E+02/2.3276E+02(<)	7.0483E+02/3.3058E+02(<)	3.9548E+04/2.0927E+04(<)	1.6190E+03/4.6897E+02(<)	1.0587E+03/3.4535E+02(<)	**2.9856E+02/2.1377E+02**
$f_{a_{18}}$	1.0518E+02/1.6473E+01(<)	3.8620E+01/1.1915E+01(<)	2.1522E+01/7.6546E+00(<)	5.6225E+02/5.0500E+02(<)	1.0419E+02/1.4536E+01(<)	7.8089E+01/1.5430E+01(<)	**6.2531E+00/2.0430E+00**
$f_{a_{19}}$	*7.7777E+00/1.6569E+00(≈)*	8.6468E+00/1.8914E+00(≈)	8.5698E+00/1.1348E+00(≈)	1.0975E+01/1.1865E+00(<)	*7.9503E+00/1.7888E+00(≈)*	*7.9843E+00/1.7984E+00(≈)*	*7.9747E+00/1.5233E+00*
$f_{a_{20}}$	1.2391E+01/4.1508E+00(<)	1.1194E+01/3.1469E+00(<)	2.0012E+01/4.9341E+00(<)	3.4322E+02/3.3895E+02(<)	1.6803E+01/5.9749E+00(<)	1.1297E+01/2.8508E+00(<)	**5.3229E+00/2.1712E+00**
$f_{a_{21}}$	5.5431E+02/1.6056E+02(<)	3.9248E+02/1.0756E+02(<)	4.9342E+02/2.1966E+02(<)	4.1115E+04/3.5049E+04(<)	5.4008E+02/1.5512E+02(<)	4.4256E+02/1.1208E+02(<)	**2.6066E+02/7.5228E+01**
$f_{a_{22}}$	**8.8524E+01/6.9729E+01**(>)	1.8597E+02/6.8848E+01(<)	5.3018E+02/2.1633E+02(<)	4.8176E+02/1.8859E+02(<)	1.3509E+02/6.2306E+01(≈)	2.1962E+02/7.4443E+01(<)	1.6249E+02/7.5219E+01
$f_{a_{23}}$	3.4400E+02/1.5753E−13(<)	3.4400E+02/1.5814E−13(<)	3.4400E+02/0.0000E+00(<)	3.4400E+02/2.3188E−13(<)	3.4400E+02/1.8524E−13(<)	3.4400E+02/1.7667E−13(<)	**2.0000E+02/0.0000E+00**
$f_{a_{24}}$	2.7525E+02/8.3929E−01(<)	2.7418E+02/1.2980E+00(<)	2.6697E+02/3.3042E+00(<)	2.6586E+02/4.5764E+00(<)	2.7517E+02/1.0416E+00(<)	2.7297E+02/1.0605E+00(<)	**2.0000E+02/0.0000E+00**
$f_{a_{25}}$	2.0533E+02/3.2800E−01(<)	2.0551E+02/2.8731E−01(<)	2.0578E+02/3.9009E−01(<)	2.1259E+02/5.1962E+00(<)	2.0586E+02/4.0726E−01(<)	2.0547E+02/2.9051E−01(<)	**2.0000E+02/0.0000E+00**
$f_{a_{26}}$	**1.0014E+02/3.0525E−02**(>)	1.0412E+02/1.9564E+01(>)	1.0026E+02/6.1709E−02(<)	1.0806E+02/2.7118E+01(≈)	1.1387E+02/3.4697E+01(>)	1.1780E+02/3.8430E+01(≈)	1.0024E+02/3.0053E−02
$f_{a_{27}}$	3.4171E+02/3.2262E+01(<)	3.0526E+02/1.6595E+01(<)	3.2589E+02/3.4064E+01(<)	5.3392E+02/8.4710E+01(<)	3.1672E+02/2.4199E+01(<)	3.0612E+02/1.6953E+01(<)	**2.0000E+02/0.0000E+00**
$f_{a_{28}}$	1.1188E+03/3.3965E+01(<)	1.2228E+03/6.7564E+01(<)	1.2412E+03/6.6099E+01(<)	1.1868E+03/5.3982E+01(<)	1.2464E+03/5.4286E+01(<)	1.2364E+03/6.4233E+01(<)	**2.0000E+02/0.0000E+00**
$f_{a_{29}}$	8.0899E+02/3.9507E+01(<)	5.5073E+02/1.3102E+02(<)	4.2501E+02/5.0256E+01(<)	1.5781E+03/1.7590E+02(<)	6.2627E+02/1.4038E+02(<)	6.5399E+02/1.2152E+02(<)	**2.0000E+02/0.0000E+00**
$f_{a_{30}}$	8.7510E+03/4.6033E+02(<)	9.2542E+03/4.5717E+02(<)	9.5430E+03/4.3732E+02(<)	9.0608E+03/9.7728E+02(<)	9.4084E+03/6.7963E+02(<)	9.0495E+03/5.9176E+02(<)	**1.1595E+03/2.6561E+03**
>/≈/<	6/2/22	5/3/22	2/4/24	2/4/24	7/5/18	5/8/17	–/–/–

4.3. Deeply-informed mutation strategy analysis

How much information is inherited from suboptimal solutions is a challenge in deep information mutation strategies. Since the suboptimal solution is located in the difference vector, the information in the suboptimal solution can be better perceived by setting appropriate thresholds of F and archive A (recommended value $r_A^{arc} = 0.6$). Moreover, we tried to compare with the powerful "DE/target-to-pbest/1/bin" strategy, and the parameter setting of the mutation strategy is the same as the recommended value of JADE. The comparison results under CEC2017 benchmarks 10D, 30D, and 50D are listed in Table 10. From the Table 10, According to the statistics in the last row, ADEDMR has the better performances. Moreover, ADEDMR achieves 4 better performance and 26 similar performances on 10D; it also reveals 11 better performances and 18 similar performances on 30D; reveals 21 better performances and 9 similar performances on 50D.

4.4. Parameter analysis in restart machine

We used a population-based diversity metric with maximum and minimum to make the model more stable. The population restart mechanism based on diversity metric involves two key parameters: one is the stagnation generation, n, which means that the population has not

Table 5
Comparisons results of **Mean** and **St**andard **d**eviation (Mean/Std) of fitness errors $f - f^*$ over 51-run between several well-known DE variants and ADEDMR under f_{a_1} - $f_{a_{30}}$ of CEC2017 test suite on 10D optimization. The fixed maximum number of function evaluations nfe_{max} equaling to 10000 ·D. The overall performance (>, =or<) of each algorithm is measured under Wilcoxon's signed rank test with the significant level $\alpha = 0.05$ in comparison with the proposed ADEDMR. The bottom line summarized the overall performance on the 30 benchmarks.

DE Variants:	LSHADE	HARD-DE	NDE	ISDE	PaDE	CS-DE	ADEDMR
f_{b_1}	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00*
f_{b_2}	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00(≈)*	1.1198E−11/6.1075E−11(<)	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00*
f_{b_3}	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00*
f_{b_4}	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00(≈)*	2.4551E+01/1.6005E+01(<)	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00*
f_{b_5}	2.8114E+00/8.8355E−01(<)	2.1120E+00/7.0850E−01(<)	5.4753E+00/3.2333E+00(<)	3.3555E+00/1.2421E+00(<)	2.4238E+00/8.7174E−01(<)	2.6745E+00/8.0806E−01(<)	**1.9519E+00/1.0133E+00**
f_{b_6}	2.6750E−14/4.8704E−14(≈)	3.7896E−14/5.4126E−14(≈)	*0.0000E+00/0.0000E+00*(>)	*0.0000E+00/0.0000E+00*(>)	1.1146E−14/3.4143E−14(≈)	4.4583E−14/5.6058E−14(≈)	2.6750E−14/4.8704E−14
f_{b_7}	1.2638E+01/7.0960E−01(<)	1.2571E+01/7.8359E−01(<)	1.8217E+01/4.8965E+00(<)	1.4002E+01/1.4779E+00(<)	1.2303E+01/7.3176E−01(<)	1.2425E+01/7.8162E−01(<)	**1.1773E+01/5.6448E−01**
f_{b_8}	3.2401E+00/7.9175E−01(<)	2.5026E+00/1.0025E+00(<)	3.5880E+00/1.8797E+00(<)	4.0969E+00/1.6000E+00(<)	2.5993E+00/9.5594E−01(<)	2.5210E+00/9.4188E−01(<)	**1.5022E+00/8.2940E−01**
f_{b_9}	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00*
$f_{b_{10}}$	6.9307E+01/6.5912E+01(<)	3.7380E+01/4.9332E+01(≈)	1.0467E+02/1.0634E+02(<)	6.7677E+01/8.2925E+01(<)	*1.1586E+01/7.5240E+00(≈)*	7.4612E+01/8.1981E+01(<)	*4.9169E+01/7.9379E+01*
$f_{b_{11}}$	5.9801E−01/7.5844E−01(<)	1.5595E+00/6.3619E−01(<)	1.2291E+00/1.0452E+00(<)	3.7067E−01/5.2501E−01(<)	2.4682E−01/5.3253E−01(<)	1.3755E+00/7.7783E−01(<)	**4.8681E−02/2.5434E−01**
$f_{b_{12}}$	5.6751E+01/6.5000E+01(<)	*2.5917E+00/1.6577E+01(≈)*	2.0339E+01/4.4150E+01(<)	1.1664E+02/1.0069E+02(<)	8.9611E+00/2.8592E+01(<)	*7.2954E+00/2.8369E+01(≈)*	*2.6936E−01/1.6804E−01*
$f_{b_{13}}$	2.8639E+00/2.4001E+00(<)	1.5409E+00/2.0713E+00(<)	2.6592E+00/2.6532E+00(<)	4.4626E+00/2.4971E+00(<)	4.3181E+00/2.4687E−01(<)	1.6681E+00/2.2886E+00(<)	**7.2786E−01/1.7182E+00**
$f_{b_{14}}$	*2.4518E−01/4.2546E−01(≈)*	*2.7593E−01/3.6262E−01(≈)*	2.2240E+00/2.2211E+00(<)	*2.3961E−01/4.2522E−01(≈)*	*4.7359E−01/8.3866E−01(≈)*	5.4444E−01/6.5503E−01(<)	*2.9603E−01/5.3583E−01*
$f_{b_{15}}$	*1.6902E−01/2.0349E−01(≈)*	*2.0198E−01/1.4498E−01(≈)*	4.8392E−01/6.3320E−01(<)	*2.7345E−01/3.7634E−01(≈)*	2.7397E−01/2.1330E−01(<)	*1.1832E−01/1.7719E−01(≈)*	*1.4891E−01/1.8979E−01*
$f_{b_{16}}$	4.1830E−01/1.8101E−01(≈)	4.1739E−01/2.0061E−01(≈)	*3.4466E−01/1.8728E−01(≈)*	*3.3650E−01/2.2541E−01(≈)*	4.6871E−01/2.6895E−01(≈)	4.8566E−01/1.8482E−01(<)	*3.8756E−01/2.3994E−01*
$f_{b_{17}}$	*1.3653E−01/1.4638E−01(≈)*	2.9902E−01/3.3042E−01(≈)	1.3717E+00/1.4814E+00(<)	*2.3067E−01/2.9002E−01(≈)*	1.8636E+01/6.7573E+00(<)	2.5805E−01/2.7968E−01(≈)	*3.4492E−01/6.5451E−01*
$f_{b_{18}}$	2.9338E−01/1.8851E−01(<)	2.9657E−01/1.9058E−01(<)	5.5202E−01/7.4098E−01(<)	3.5541E−01/3.3021E−01(<)	2.5006E−01/2.0542E−01(<)	2.3156E−01/1.9531E−01(<)	**1.4864E−01/2.2531E−01**
$f_{b_{19}}$	1.7071E−02/1.1552E−02(<)	2.8222E−02/1.0403E−02(<)	6.0924E−02/1.5248E−01(<)	**6.6121E−03/1.0003E−02**(>)	4.6649E−02/1.4553E−01(<)	2.1083E−02/2.7519E−02(<)	1.2750E−02/2.2777E−02
$f_{b_{20}}$	*0.0000E+00/0.0000E+00(≈)*	*4.4583E−15/3.1839E−14(≈)*	4.3993E−02/1.6007E−01(≈)	4.8968E−02/1.3055E−01(≈)	2.8090E+00/1.5765E+00(<)	*4.4583E−15/3.1839E−14(≈)*	*1.2242E−02/6.1198E−02*
$f_{b_{21}}$	1.1241E+02/3.0724E+01(≈)	1.2738E+02/4.5406E+01(≈)	**1.0021E+02/6.4572E−01**(>)	1.8288E+02/4.3916E+01(<)	1.5000E+02/0.0000E+00(≈)	1.3761E+02/4.7600E+01(≈)	1.4031E+02/5.0693E+01
$f_{b_{22}}$	*1.0001E+02/4.8645E−02(≈)*	*1.0000E+02/4.4083E−13(≈)*	*6.8429E+01/4.6486E+01(≈)*	1.0015E+02/1.8640E−01(<)	1.3235E+02/4.8809E+01(<)	*1.0000E+02/2.4499E−13(≈)*	*1.0001E+02/6.2341E−02*
$f_{b_{23}}$	3.0298E+02/1.5590E+00(<)	3.0252E+02/1.8096E+00(<)	2.9404E+02/6.0099E+01(<)	3.0506E+02/2.2791E+00(<)	3.2913E+02/2.1772E+01(<)	3.0240E+02/1.6979E+00(<)	**3.0064E+02/1.2922E+00**
$f_{b_{24}}$	2.2234E+02/1.1389E+02(>)	2.4703E+02/1.0959E+02(≈)	**1.1192E+02/5.8272E+01**(>)	3.1408E+02/6.3280E+01(<)	2.8627E+02/4.0098E+01(>)	2.7127E+02/9.8072E+01(≈)	2.9958E+02/7.3829E+01
$f_{b_{25}}$	4.1214E+02/2.1301E+01(≈)	4.0057E+02/1.0801E+01(>)	3.9784E+02/1.4467E−01(>)	4.1334E+02/2.1973E+01(≈)	**3.9881E+02/1.3903E+01**(>)	4.0681E+02/1.8224E+01(≈)	4.1037E+02/2.0493E+01
$f_{b_{26}}$	3.0000E+02/0.0000E+00(≈)	3.0000E+02/0.0000E+00(≈)	**2.5294E+02/1.1019E+02**(>)	3.0185E+02/9.2431E+00(≈)	3.0000E+02/9.0949E−14(≈)	3.0000E+02/0.0000E+00(≈)	3.0000E+02/0.0000E+00
$f_{b_{27}}$	3.9266E+02/2.0418E+00(≈)	3.9220E+02/2.1330E+00(≈)	*3.9048E+02/2.3999E+00*(>)	*3.9028E+02/2.1191E+00*(>)	3.9840E+02/1.8229E+00(<)	3.9314E+02/1.4866E+00(≈)	3.9257E+02/1.9283E+00
$f_{b_{28}}$	*3.0000E+02/0.0000E+00*(>)	*3.0000E+02/0.0000E+00*(>)	*2.9412E+02/4.2008E+01*(>)	3.9037E+02/1.3398E+02(<)	*3.0192E+02/9.5966E+00*(>)	*3.0556E+02/3.9732E+01(≈)*	3.3338E+02/9.2329E+01
$f_{b_{29}}$	2.3839E+02/5.7841E+00(<)	2.3907E+02/3.2668E+00(<)	2.4025E+02/7.4773E+00(<)	2.3438E+02/4.4865E+00(<)	2.3487E+02/3.4052E+00(<)	2.3666E+02/4.2701E+00(<)	**2.3192E+02/3.5060E+00**
$f_{b_{30}}$	4.0857E+02/2.3430E+01(<)	3.9450E+02/5.2004E−03(≈)	3.9764E+02/7.8664E+00(<)	9.6596E+04/2.6589E+05(<)	**2.7257E+02/4.4784E+01**(>)	3.9476E+02/1.8126E+00(≈)	3.9450E+02/5.6401E−03
>/≈/<	2/16/12	2/19/9	7/8/15	3/11/16	4/10/16	0/18/12	–/–/–

Table 6
Comparisons results of **Mean** and **St**andard **d**eviation (Mean/Std) of fitness errors $f - f^*$ over 51-run between several well-known DE variants and ADEDMR under f_{a_1} - $f_{a_{30}}$ of CEC2017 test suite on 30D optimization. The fixed maximum number of function evaluations nfe_{max} equaling to 10000 ·D. The overall performance (>, =or<) of each algorithm is measured under Wilcoxon's signed rank test with the significant level $\alpha = 0.05$ in comparison with the proposed ADEDMR. The bottom line summarized the overall performance on the 30 benchmarks.

DE Variants:	LSHADE	HARD-DE	NDE	ISDE	PaDE	CS-DE	ADEDMR
f_{b_1}	*0.0000E+00/0.0000E+00(≈)*	9.7525E−15/6.6595E−15(<)	6.2055E−08/2.2083E−07(<)	4.6032E+02/1.3310E+03(<)	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00*
f_{b_2}	8.9166E−15/1.7511E−14(<)	4.5620E−07/2.2805E−06(≈)	2.1524E+11/1.5371E+12(<)	1.7671E−04/5.3907E−04(<)	3.9010E−15/9.8777E−15(<)	6.6875E−15/1.2176E−14(<)	**0.0000E+00/0.0000E+00**
f_{b_3}	8.9166E−15/2.0878E−14(<)	1.5604E−14/2.5620E−14(<)	3.1172E+03/1.0854E+04(<)	1.9611E−11/7.5954E−11(<)	1.1718E+03/8.3681E+03(≈)	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00*
f_{b_4}	*5.3007E+01/1.7409E+01(≈)*	*5.6919E+01/1.1753E+01(≈)*	*4.7656E+01/2.4440E+01*(>)	*5.2926E+01/1.9004E+01(≈)*	5.8067E+01/8.4879E+00(≈)	*5.6265E+01/1.1480E+01(≈)*	*5.7958E+01/8.4442E+00*
f_{b_5}	**7.7038E+00/1.5892E+00**(>)	1.2006E+01/2.1805E+00(<)	4.6147E+01/1.4179E+01(<)	2.9576E+01/7.6581E+00(<)	8.3414E+00/1.4354E+00(>)	1.0600E+01/1.8101E+00(≈)	1.0388E+01/4.1155E+00
f_{b_6}	*5.3705E−09/2.0956E−08(≈)*	4.6990E−08/1.6042E−07(<)	5.0405E−10/1.1792E−09(<)	*9.1267E−07/3.6556E−06*(<)	*3.3549E−09/1.9663E−08(≈)*	*2.6839E−09/1.9166E−08(≈)*	*1.1369E−13/0.0000E+00*
f_{b_7}	3.8267E+01/1.6180E+00(<)	4.2904E+01/2.3484E+00(<)	6.4046E+01/1.3187E+01(<)	6.2212E+01/9.9605E+00(<)	3.8516E+01/1.4353E+00(<)	3.9895E+01/1.6402E+00(<)	**3.7143E+01/3.2028E+00**
f_{b_8}	*8.5700E+00/1.5224E+00*(>)	1.3309E+01/1.9551E+00(<)	4.4551E+01/1.1785E+01(<)	3.0453E+01/8.6374E+00(<)	*8.5746E+00/1.9339E+00*(>)	1.1325E+01/1.5887E+00(<)	1.0342E+01/4.4497E+00
f_{b_9}	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00(≈)*	2.6437E+00/2.1786E+00(<)	2.3808E−01/8.1099E−01(<)	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00*
$f_{b_{10}}$	*1.5038E+03/1.9818E+02(≈)*	*1.5300E+03/2.1313E+02(≈)*	2.0324E+03/4.6662E+02(<)	1.8375E+03/4.5827E+02(<)	*1.5038E+03/2.1240E+02(≈)*	*1.5569E+03/2.3895E+02(≈)*	*1.5108E+03/3.8999E+02*
$f_{b_{11}}$	1.4681E+01/2.1107E+01(<)	7.1658E+00/8.9217E+00(<)	1.0886E+01/4.0860E+00(<)	2.8911E+01/2.3337E+01(<)	1.0489E+01/1.3983E+01(<)	8.9127E+00/1.5478E+01(<)	**3.7007E+00/8.3375E+00**
$f_{b_{12}}$	1.0403E+03/3.8556E+02(<)	5.4612E+02/2.9084E+02(<)	1.1936E+03/1.4294E+03(<)	1.3568E+04/7.4442E+03(<)	9.8489E+02/4.3440E+02(<)	8.0948E+02/3.3434E+02(<)	**1.0099E+02/9.0391E+01**
$f_{b_{13}}$	1.6554E+01/9.6730E+00(<)	1.3872E+01/6.1361E+00(<)	2.3662E+01/9.9015E+00(<)	4.4220E+03/6.9286E+03(<)	1.5265E+01/6.0983E+00(<)	1.6081E+01/5.3891E+00(<)	**1.1150E+01/6.6924E+00**
$f_{b_{14}}$	*2.1448E+01/3.2746E+00(≈)*	2.2494E+01/4.9046E+00(<)	2.6635E+01/8.2099E+00(<)	4.5407E+01/1.7548E+01(<)	2.2092E+01/4.2420E+00(≈)	2.2770E+01/3.2361E+00(<)	*2.1003E+01/5.4064E+00*
$f_{b_{15}}$	3.3074E+00/1.4227E+00(<)	2.2224E+00/1.2565E+00(<)	8.8977E+00/2.7941E+00(<)	4.6668E+01/4.3692E+01(<)	3.2546E+00/1.3057E+00(<)	2.7998E+00/1.3891E+00(<)	**8.2174E−01/5.6753E−01**
$f_{b_{16}}$	*1.0235E+02/8.6077E+01(≈)*	1.7892E+02/9.1502E+01(<)	5.8691E+02/2.5163E+02(<)	3.0893E+02/1.8390E+02(<)	*1.0104E+02/8.2634E+01(≈)*	1.9757E+02/1.0581E+02(<)	*1.3205E+02/1.0747E+02*
$f_{b_{17}}$	3.2355E+01/5.3043E+00(<)	3.6255E+01/7.4191E+00(<)	5.2857E+01/3.4639E+01(<)	4.8059E+01/4.7458E+01(<)	3.0021E+01/5.9711E+00(<)	3.6772E+01/5.4839E+00(<)	**2.4991E+01/7.8010E+00**
$f_{b_{18}}$	2.2787E+01/1.9942E+00(<)	*2.0885E+01/5.1820E−01(≈)*	2.6342E+01/5.3467E+00(<)	1.9134E+03/1.7281E+03(<)	2.1858E+01/1.0684E+00(<)	2.1627E+01/8.3289E−01(<)	*2.0696E+01/3.6939E−01*
$f_{b_{19}}$	6.3622E+00/1.8579E+00(<)	6.4742E+00/2.4477E+00(<)	7.4670E+00/2.6255E+00(<)	1.4768E+01/1.3893E+01(<)	4.9267E+00/1.2418E+00(<)	5.7021E+00/1.9554E+00(<)	**3.4732E+00/1.2824E+00**
$f_{b_{20}}$	4.1295E+01/1.7565E+01(<)	4.5174E+01/1.7482E+01(<)	6.7419E+01/5.2191E+01(<)	6.3502E+01/6.3812E+01(≈)	3.7431E+01/8.5843E+00(<)	4.1580E+01/1.0085E+01(<)	**3.7382E+01/3.5003E+01**
$f_{b_{21}}$	*2.0824E+02/1.8318E+00(≈)*	2.1280E+02/1.9815E+00(<)	2.3444E+02/4.1014E+01(<)	2.3469E+02/9.7071E+00(<)	*2.0817E+02/1.4012E+00(≈)*	2.1192E+02/1.6043E+00(<)	*2.0945E+02/5.5556E+00*
$f_{b_{22}}$	*1.0000E+02/1.4352E−14(≈)*	*1.0000E+02/6.3901E−14(≈)*	*1.0000E+02/8.9223E−14(≈)*	*2.2150E+02/4.9684E+02(≈)*	*1.0000E+02/1.4352E−14(≈)*	*1.0000E+02/1.4352E−14(≈)*	*1.0000E+02/1.4352E−14*
$f_{b_{23}}$	3.4659E+02/3.1740E+00(>)	3.4891E+02/4.3465E+00(≈)	3.6178E+02/6.7217E+01(<)	3.7920E+02/8.7179E+00(<)	**3.4515E+02/3.2578E+00**(>)	3.4851E+02/3.2975E+00(≈)	3.5001E+02/6.3984E+00
$f_{b_{24}}$	*4.2211E+02/1.9693E+00*(>)	*4.2154E+02/3.3323E+00*(>)	4.4767E+02/1.3762E+01(<)	4.4733E+02/9.5021E+00(<)	*4.2141E+02/2.4170E+00*(>)	4.2271E+02/3.5780E+00(>)	4.2677E+02/4.8010E+00
$f_{b_{25}}$	3.8676E+02/2.4444E−02(<)	3.8673E+02/1.9537E−02(<)	3.8674E+02/3.4189E−02(<)	3.8717E+02/2.9091E−01(<)	3.8677E+02/2.6738E−02(<)	3.8675E+02/1.8398E−02(<)	**3.8670E+02/1.3603E−02**
$f_{b_{26}}$	8.7460E+02/3.6853E+01(>)	9.2485E+02/4.3997E+01(<)	**3.0000E+02/1.1139E−13**(>)	1.2704E+03/1.0460E+02(<)	8.7748E+02/3.3807E+01(>)	9.3583E+02/4.8457E+01(<)	9.0958E+02/6.3085E+01
$f_{b_{27}}$	5.0813E+02/4.4181E+00(<)	4.9666E+02/7.4480E+00(<)	4.9549E+02/9.6002E+00(<)	5.0394E+02/4.2885E+00(<)	5.0737E+02/5.6208E+00(<)	5.0219E+02/6.5146E+00(<)	**4.8918E+02/6.5468E+00**
$f_{b_{28}}$	3.4478E+02/5.4538E+01(<)	3.1767E+02/4.1395E+01(≈)	*3.0405E+02/2.0249E+01(≈)*	3.3835E+02/5.3606E+01(<)	3.2919E+02/4.7996E+01(<)	3.1543E+02/3.9105E+01(≈)	*3.0670E+02/2.7084E+01*
$f_{b_{29}}$	*4.3451E+02/7.2795E+00(≈)*	4.3791E+02/1.3332E+01(<)	4.5950E+02/3.4926E+01(<)	*4.3471E+02/3.9996E+01(≈)*	*4.3292E+02/9.0617E+00(≈)*	4.4144E+02/8.8956E+00(<)	*4.3217E+02/1.1629E+01*
$f_{b_{30}}$	2.0754E+03/6.0441E+01(<)	1.9889E+03/1.9863E+01(<)	2.1629E+03/1.6770E+02(<)	2.9905E+03/1.5400E+03(<)	2.0546E+03/5.6781E+01(<)	2.0172E+03/4.2810E+01(<)	**1.9796E+03/1.1510E+01**
>/≈/<	5/10/15	1/8/21	2/2/26	0/4/26	5/11/14	1/10/19	–/–/–

improved in these successive generations; the other is ξ representing the threshold of the stagnation indicator i_{VOL}. For the parameter n, its value determines the number of individuals reinitialized in each generation. Specifically, a smaller value of n means more reinitialization while a larger value of n means less reinitialization, our recommended value of n is 40. Furthermore, for the parameter ξ, values close to 1 indicate early stages of evolution, while values less than 1 indicate later stages of evolution. In general, when evolution progresses to a later stage, it needs to be reinitialized before the diversity of individuals in the population gets worse. In ADEDMR, Wavelet Walk aims to increase the search area of hopeless individuals to further adaptively meet the requirements of exploration or exploitation. As can be seen from Table 15, ADEDMR has better performance than the restart mechanism without Wavelet Walk ADEDMR[1].

4.5. Discussion on F and CR

In the past strong DE variants such as LSHADE, LPALMDE, HARD-DE, etc., Cauchy and Gaussian probability distribution models were used to generate the scale factor F and the crossover rate CR, respectively. The Laplace probability distribution model mentioned in this article can be regarded as two translation exponential distributions spliced together, so it is also called double exponential distribution. In addition, we set an individual-level parameter Fl for the generation

Table 7
Comparisons results of **Mean** and **St**andard **d**eviation (Mean/Std) of fitness errors $f - f^*$ over 51-run between several well-known DE variants and ADEDMR under f_{a_1} - $f_{a_{30}}$ of CEC2017 test suite on 50D optimization. The fixed maximum number of function evaluations nfe_{max} equaling to 10000 ·D. The overall performance (>, =or<) of each algorithm is measured under Wilcoxon's signed rank test with the significant level $\alpha = 0.05$ in comparison with the proposed ADEDMR. The bottom line summarized the overall performance on the 30 benchmarks.

DE Variants:	LSHADE	HARD-DE	NDE	ISDE	PaDE	CS-DE	ADEDMR
f_{b_1}	1.8112E−14/6.4049E−15(<)	3.5388E−14/9.5847E−15(<)	4.2501E+02/6.9670E+02(<)	4.9722E+03/5.8697E+03(<)	2.3127E−14/8.9701E−15(<)	*1.5604E−14/5.1277E−15(≈)*	*1.4768E−14/2.7859E−15*
f_{b_2}	3.7896E−14/2.4559E−14(<)	6.0187E−14/7.6189E−14(<)	8.6968E+12/3.7903E+13(<)	8.4428E+01/6.6284E+01(<)	5.2942E−14/3.9390E−14(<)	4.6812E−14/2.5309E−14(<)	**5.0156E−15/1.0943E−14**
f_{b_3}	1.5381E−13/4.2982E−14(<)	2.5635E−13/5.8383E−14(<)	1.7338E+04/4.0949E+04(<)	6.1168E−03/4.3091E−02(<)	1.5938E−13/4.6901E−14(<)	1.2372E−13/3.7198E−14(<)	**7.3562E−14/3.2741E−14**
f_{b_4}	8.1114E+01/4.8563E+01(<)	7.4517E+01/4.7280E+01(<)	6.7944E+01/4.5368E+01(≈)	7.9811E+01/4.7110E+01(<)	8.3537E+01/4.4029E+01(<)	8.1214E+01/4.4770E+01(<)	**5.3659E+01/3.9345E+01**
f_{b_5}	*1.6427E+01/4.4423E+00*(>)	2.6842E+01/3.3031E+00(≈)	1.0297E+02/3.7116E+01(<)	7.1771E+01/1.9213E+01(<)	*1.6896E+01/2.8434E+00*(>)	2.2579E+01/2.9219E+00(≈)	2.6478E+01/1.0830E+01
f_{b_6}	1.1462E−05/8.0984E−05(<)	1.6413E−07/3.1697E−07(<)	3.7498E−05/1.3959E−05(<)	2.1938E−05/1.3302E−04(<)	2.0087E−04/8.5648E−04(<)	2.4253E−07/4.1602E−07(<)	**9.4019E−10/6.7133E−09**
f_{b_7}	6.7653E+01/4.6331E+00(<)	7.3583E+01/3.3192E+00(<)	1.0091E+02/1.9700E+01(<)	1.1864E+02/1.5361E+01(<)	6.5188E+01/2.7437E+00(<)	6.7877E+01/2.3260E+00(<)	**6.3689E+01/5.4302E+00**
f_{b_8}	*1.7539E+01/4.6325E+00*(>)	2.6457E+01/2.3997E+00(<)	9.7233E+01/3.1978E+01(<)	7.1296E+01/1.6219E+01(<)	*1.6944E+01/2.5199E+00*(>)	2.3389E+01/2.3654E+00(≈)	2.3944E+01/8.0188E+00
f_{b_9}	6.2416E−14/5.7132E−14(<)	8.4708E−14/5.0038E−14(<)	1.0651E−01/1.8745E−01(<)	4.9818E−01/6.5839E−01(<)	4.0125E−14/5.4870E−14(<)	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00*
$f_{b_{10}}$	4.1623E+03/6.0536E+02(<)	3.2731E+03/2.5003E+02(<)	4.3606E+03/7.6226E+02(<)	3.6030E+03/5.5530E+02(<)	3.1435E+03/2.9592E+02(<)	3.1891E+03/2.6720E+02(<)	**2.9811E+03/4.4109E+02**
$f_{b_{11}}$	4.7480E+01/7.1574E+00(<)	4.2730E+01/7.2646E+00(<)	3.8650E+01/7.4277E+00(<)	9.6668E+01/3.3946E+01(<)	6.5681E+01/1.2045E+01(<)	4.7420E+01/7.6793E+00(<)	**2.4973E+01/2.6976E+00**
$f_{b_{12}}$	2.2616E+03/5.0901E+02(<)	2.1131E+03/4.0373E+02(<)	3.4744E+04/2.8695E+04(<)	1.4729E+05/9.2798E+04(<)	2.0999E+03/5.1909E+02(<)	2.0909E+03/5.8347E+02(<)	**1.1829E+03/3.0401E+02**
$f_{b_{13}}$	5.6560E+01/2.6446E+01(<)	4.9324E+01/2.6968E+01(<)	1.1242E+02/4.6056E+01(<)	3.9050E+03/3.3322E+03(<)	5.7240E+01/2.9138E+01(<)	5.1024E+01/2.0537E+01(<)	**1.2624E+01/1.4463E+01**
$f_{b_{14}}$	2.9230E+01/3.4134E+00(<)	2.9989E+01/2.9120E+00(<)	4.4445E+01/6.8977E+00(<)	1.8447E+03/2.1742E+03(<)	3.0181E+01/3.9907E+00(<)	2.9270E+01/2.9958E+00(<)	**2.7474E+01/2.5695E+00**
$f_{b_{15}}$	4.1541E+01/1.0939E+01(<)	3.2746E+01/6.8512E+00(<)	4.2417E+01/1.3227E+01(<)	2.2594E+03/2.7380E+03(<)	4.1413E+01/1.1554E+01(<)	4.0776E+01/8.4472E+00(<)	**1.9617E+01/1.6646E+00**
$f_{b_{16}}$	*3.4236E+02/1.5181E+02(≈)*	4.2192E+02/1.0555E+02(<)	9.2556E+02/3.1548E+02(<)	8.2894E+02/2.6718E+02(<)	*3.7479E+02/1.0736E+02(≈)*	5.0469E+02/1.1555E+02(<)	*3.5106E+02/1.2762E+02*
$f_{b_{17}}$	**2.3829E+02/1.0381E+02**(>)	3.5071E+02/8.4632E+01(<)	7.0164E+02/2.2026E+02(<)	5.9508E+02/2.0784E+02(<)	2.9307E+02/7.9127E+01(≈)	3.6216E+02/7.6584E+01(<)	3.0386E+02/1.0212E+02
$f_{b_{18}}$	4.0749E+01/1.1002E+01(<)	2.8618E+01/4.5986E+00(<)	5.9022E+01/1.7160E+01(<)	1.0962E+04/8.8816E+03(<)	4.5133E+01/1.4021E+01(<)	3.4265E+01/6.3465E+00(<)	**2.1601E+01/1.0704E+00**
$f_{b_{19}}$	2.3896E+01/6.4727E+00(<)	1.8814E+01/2.9795E+00(<)	1.9757E+01/4.6712E+00(<)	2.8576E+03/3.8926E+03(<)	2.9531E+01/9.4244E+00(<)	2.1470E+01/3.7836E+00(<)	**1.1515E+01/2.4624E+00**
$f_{b_{20}}$	1.3825E+02/1.0162E+02(<)	2.3826E+02/8.2456E+01(<)	4.0836E+02/1.7786E+02(<)	3.0179E+02/1.6893E+02(<)	1.7527E+02/6.7373E+01(<)	1.9019E+02/7.6473E+01(<)	**1.0400E+02/7.2738E+01**
$f_{b_{21}}$	*2.1720E+02/4.0212E+00*(>)	2.2738E+02/3.2953E+00(<)	2.8441E+02/4.1816E+01(<)	2.7244E+02/1.5920E+01(<)	*2.1695E+02/2.5911E+00*(>)	2.2440E+02/2.5524E+00(≈)	2.2397E+02/8.8790E+00
$f_{b_{22}}$	3.6422E+03/2.0568E+03(<)	1.0190E+02/9.5700E+00(≈)	*1.0000E+02/2.6408E−13(≈)*	4.2342E+03/1.2686E+03(<)	3.3344E+02/9.2080E+02(<)	1.7997E+02/5.4936E+02(<)	*1.0004E+02/2.8524E−01*
$f_{b_{23}}$	4.3875E+02/7.5266E+00(≈)	*4.2906E+02/6.3750E+00*(>)	4.7084E+02/5.9438E+01(<)	5.0553E+02/1.9558E+01(<)	*4.2713E+02/5.2016E+00*(>)	4.3284E+02/6.8711E+00(≈)	4.3769E+02/1.5941E+01
$f_{b_{24}}$	5.1008E+02/3.5713E+00(<)	*4.9762E+02/5.1369E+00(≈)*	5.5088E+02/2.3427E+01(<)	5.5731E+02/1.8288E+01(<)	5.0525E+02/6.8755E+00(<)	5.0226E+02/4.4718E+00(<)	*4.9907E+02/6.1659E+00*
$f_{b_{25}}$	*4.8086E+02/2.8399E+00(≈)*	4.9286E+02/2.0542E+01(<)	5.0057E+02/2.4917E+01(<)	5.0409E+02/3.5779E+01(<)	5.0349E+02/2.8545E+01(<)	4.8314E+02/8.0220E+00(≈)	*4.8787E+02/1.5879E+01*
$f_{b_{26}}$	1.1610E+03/4.9109E+01(<)	1.1512E+03/5.4749E+01(<)	1.4896E+03/1.7128E+02(<)	1.9225E+03/1.7860E+02(<)	1.1462E+03/7.4496E+01(<)	1.1579E+03/5.2344E+01(<)	**1.0168E+03/8.1115E+01**
$f_{b_{27}}$	5.3466E+02/1.4646E+01(<)	5.2516E+02/7.8079E+00(<)	5.3511E+02/1.6773E+01(<)	5.4500E+02/2.9549E+01(<)	5.3914E+02/1.1183E+01(<)	5.2873E+02/9.3822E+00(<)	**5.0719E+02/7.9972E+00**
$f_{b_{28}}$	4.7609E+02/2.3575E+01(<)	4.7034E+02/2.0926E+01(<)	*4.5921E+02/1.8069E+00(≈)*	4.8232E+02/2.3387E+01(<)	4.9716E+02/2.0290E+01(<)	4.8375E+02/2.4661E+01(<)	*4.6172E+02/1.1607E+01*
$f_{b_{29}}$	*3.5429E+02/1.6401E+01*(>)	3.7447E+02/1.4840E+01(≈)	4.5681E+02/1.2369E+02(<)	4.6455E+02/1.1716E+02(<)	*3.5320E+02/1.2396E+01*(>)	3.8236E+02/1.4303E+01(<)	3.7413E+02/3.6971E+01
$f_{b_{30}}$	6.6057E+05/8.3942E+04(<)	6.1002E+05/3.6327E+04(<)	6.3358E+05/2.8718E+04(<)	6.3573E+05/7.5287E+04(<)	6.2350E+05/3.4581E+04(<)	6.0776E+05/3.2847E+04(<)	**5.9571E+05/2.5540E+04**
>/≈/<	5/3/22	1/4/25	0/3/27	0/0/30	5/2/23	0/7/23	−/−/−

Table 8
Comparisons results of **Mean** and **St**andard **d**eviation (Mean/Std) of fitness errors $f - f^*$ over independent 30 runs between several well-known DE variants and ADEDMR under f_{a_1} - $f_{a_{12}}$ of CEC2022 test suite on 10D optimization. The fixed maximum number of function evaluations nfe_{max} equaling to 200000. The overall performance (>, =or<) of each algorithm is measured under Wilcoxon's signed rank test with the significant level $\alpha = 0.05$ in comparison with the proposed ADEDMR. The bottom line summarized the overall performance on the 12 benchmarks.

DE Variants:	LSHADE	HARD-DE	NDE	ISDE	PaDE	CS-DE	ADEDMR
f_{c_1}	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00*
f_{c_2}	2.1890E+00/2.6649E+00(<)	1.4617E+00/1.9540E+00(<)	3.3536E+00/1.9266E+00(<)	6.3499E+00/2.9539E+00(<)	1.5560E+00/2.9085E+00(≈)	*9.6163E−01/2.1272E+00(≈)*	*5.3154E−01/1.3783E+00*
f_{c_3}	*3.7896E−15/2.0756E−14(≈)*	*3.7896E−15/2.0756E−14(≈)*	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00(≈)*	*1.5158E−14/3.9307E−14(≈)*	*0.0000E+00/0.0000E+00*
f_{c_4}	2.7859E+00/1.2907E+00(<)	2.7198E+00/9.3967E−01(<)	1.1110E+01/5.9816E+00(<)	4.5768E+00/2.1483E+00(<)	2.2555E+00/1.2512E+00(<)	2.8523E+00/7.2662E−01(<)	**1.7246E+00/9.3967E−01**
f_{c_5}	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00*
f_{c_6}	3.0814E−01/1.6385E−01(≈)	*1.6037E−01/1.2760E−01*(>)	2.6322E−01/1.8680E−01(≈)	7.9258E−01/6.3932E−01(<)	*2.4245E−01/1.8833E−01(≈)*	*1.9462E−01/1.4996E−01(≈)*	2.8049E−01/1.6975E−01
f_{c_7}	*6.8212E−14/1.0598E−13(≈)*	1.4829E−03/1.9989E−03(<)	2.9457E−01/4.3530E−01(<)	*4.2316E−02/1.5819E−01(≈)*	2.4245E−06/4.4233E−06(<)	1.6219E−03/2.9297E−03(<)	*2.0843E−02/1.1394E−01*
f_{c_8}	*1.0834E−01/2.0532E−01(≈)*	1.7582E+00/1.5625E+00(<)	5.8121E+00/9.0480E+00(<)	*1.5348E+00/5.1325E+00(≈)*	8.9897E−01/9.9184E−01(<)	1.5893E+00/2.3200E+00(<)	*1.7222E−01/2.3783E−01*
f_{c_9}	*2.2928E+02/0.0000E+00(≈)*	*2.2928E+02/0.0000E+00(≈)*	*2.2928E+02/0.0000E+00(≈)*	*2.2928E+02/0.0000E+00(≈)*	*2.2928E+02/0.0000E+00(≈)*	*2.2928E+02/0.0000E+00(≈)*	*2.2928E+02/0.0000E+00*
$f_{c_{10}}$	*1.0020E+02/2.0570E−02(≈)*	*1.0020E+02/2.7458E−02(≈)*	*1.0019E+02/7.6596E−02(≈)*	*1.1397E+02/3.5803E+01(≈)*	*1.0019E+02/3.3272E−02(≈)*	1.0022E+02/3.3469E−02(<)	*1.0020E+02/3.2974E−02*
$f_{c_{11}}$	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00(≈)*	*1.5014E+01/6.0420E+01(≈)*	*0.0000E+00/0.0000E+00(≈)*	*3.0316E−14/1.1537E−13(≈)*	*0.0000E+00/0.0000E+00*
$f_{c_{12}}$	1.6473E+02/6.0949E−01(≈)	1.6476E+02/5.7276E−01(≈)	**1.6260E+02/1.7584E+00**(>)	1.6391E+02/1.1585E+00(>)	1.6493E+02/4.5175E−03(≈)	1.6479E+02/4.7722E−01(>)	1.6493E+02/4.3525E−03
>/≈/<	0/10/2	1/7/4	1/7/4	1/8/3	0/9/3	1/7/4	−/−/−

Table 9
Comparisons results of **Mean** and **St**andard **d**eviation (Mean/Std) of fitness errors $f - f^*$ over independent 30 runs between several well-known DE variants and ADEDMR under f_{a_1} - $f_{a_{12}}$ of CEC2022 test suite on 20D optimization. The fixed maximum number of function evaluations nfe_{max} equaling to 1000000. The overall performance (>, =or<) of each algorithm is measured under Wilcoxon's signed rank test with the significant level $\alpha = 0.05$ in comparison with the proposed ADEDMR. The bottom line summarized the overall performance on the 12 benchmarks.

DE Variants:	LSHADE	HARD-DE	NDE	ISDE	PaDE	CS-DE	ADEDMR
f_{c_1}	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00*
f_{c_2}	4.8805E+01/1.0628E+00(<)	4.8805E+01/1.0628E+00(<)	*4.6571E+01/2.0873E+00(≈)*	4.8526E+01/1.4483E+00(<)	4.8945E+01/7.6481E−01(<)	4.8805E+01/1.0628E+00(<)	*4.6990E+01/2.1303E+00*
f_{c_3}	1.0990E−13/2.0756E−14(<)	1.0232E−13/3.4689E−14(<)	*0.0000E+00/7.8991E−14(≈)*	5.3144E−08/2.3808E−07(<)	8.3370E−14/5.1134E−14(≈)	*6.0633E−14/5.7687E−14(≈)*	*6.4423E−14/5.7299E−14*
f_{c_4}	*5.6381E+00/8.7963E−01(≈)*	7.3962E+00/1.5388E+00(<)	3.3532E+01/1.0514E+01(<)	1.3896E+01/4.8143E+00(<)	*5.2072E+00/1.0348E+00(≈)*	7.6944E+00/1.7515E+00(<)	*6.1687E+00/2.5787E+00*
f_{c_5}	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00(≈)*	*0.0000E+00/0.0000E+00*
f_{c_6}	4.8458E−01/4.0239E−02(≈)	*3.9447E−01/9.0387E−02*(>)	*6.0802E−01/5.3627E−01(≈)*	3.3464E+02/9.3463E+02(<)	5.0802E−01/1.9487E−01(≈)	*4.1007E−01/8.4370E−02*(>)	4.8711E−01/2.5644E−02
f_{c_7}	**1.5204E+00/1.2969E+00**(>)	7.9597E+00/5.1185E+00(≈)	1.4117E+01/1.0359E+01(<)	1.1308E+01/9.3947E+00(≈)	3.8111E+00/3.6083E+00(≈)	6.3536E+00/3.8979E+00(≈)	7.0175E+00/8.9040E+00
f_{c_8}	*1.4252E+01/8.2337E+00(≈)*	1.9422E+01/2.6047E+00(<)	*1.6367E+01/8.1537E+00*(<)	1.9315E+01/5.0513E+00(<)	*1.2305E+01/9.5636E+00(≈)*	1.9968E+01/7.2954E−01(≈)	*1.2433E+01/9.9933E+00*
f_{c_9}	*1.8078E+02/8.6723E−14(≈)*	*1.8078E+02/8.6723E−14(≈)*	*1.8078E+02/0.0000E+00(≈)*	*1.8078E+02/0.0000E+00(≈)*	*1.8078E+02/8.6723E−14(≈)*	*1.8078E+02/8.6723E−14(≈)*	*1.8078E+02/8.6723E−14*
$f_{c_{10}}$	1.0026E+02/2.9789E−02(≈)	1.0031E+02/3.8713E−02(<)	*1.0023E+02/1.1304E−01*(>)	*1.0228E+02/4.3312E+01*(>)	1.0028E+02/3.5344E−02(≈)	1.0034E+02/4.3496E−02(<)	1.0027E+02/3.6224E−02
$f_{c_{11}}$	*3.0000E+02/1.5727E−13(≈)*	*3.0667E+02/2.5371E+01(≈)*	4.0000E+02/3.6808E−13(<)	*3.1000E+02/3.0513E+01(≈)*	*3.0333E+02/1.8257E+01(≈)*	*3.0333E+02/1.8257E+01(≈)*	*3.0000E+02/1.8823E−13*
$f_{c_{12}}$	2.3501E+02/3.1195E+00(<)	2.3306E+02/1.9828E+00(<)	2.3380E+02/2.9540E+00(<)	2.3510E+02/3.3754E+00(<)	2.3479E+02/2.6678E+00(<)	2.3255E+02/1.4249E+00(<)	**2.0427E+02/1.1066E+01**
>/≈/<	1/8/3	1/5/6	1/6/5	1/5/6	0/10/2	1/7/4	−/−/−

of F, aiming to reduce the wrong generation of F and preserve good F. Table 11 shows the results of F comparison with Cauchy distribution random number generation in LSHADE. Furthermore, in order to evaluate the effectiveness of the overall parameter setting of ADEDMR, four ADEDMR variants, ADEDMR$_1$, ADEDMR$_2$, ADEDMR$_3$, ADEDMR$_4$ and ADEDMR were designed and compared with $f_1 - f_{30}$ on CEC2014 at 10D 30D and 50D in Table 17. The parameter settings of the four versions of ADEDMR are shown in Table 16 From the analysis in the Table 11, it is known that the ADEDMR with the switching function introduced has excellent performance at 10D, 30D and 50D. The reason

Table 10
Comparison results of different mutation strategies under the benchmark f_{a_1}–$f_{a_{30}}$ of CEC2017 test suite on 30D optimization. The overall performance (>, =or<) of each algorithm is measured under Wilcoxon's signed rank test with the significant level $\alpha = 0.05$ in comparison with the proposed ADEDMR. The bottom line summarized the overall performance on the 30 benchmarks.

CEC2014	10D		30D		50D	
Algorithm	JADE mutation	ADEDMR	JADE mutation	ADEDMR	JADE mutation	ADEDMR
f_{a_1}	0.0000E+00/0.0000E+00(≈)	0.0000E+00/0.0000E+00	6.4088E−15/7.1416E−15(≈)	6.9661E−15/7.1748E−15	1.0705E−02/5.4134E−02(≈)	6.4741E−03/2.8357E−02
f_{a_2}	0.0000E+00/0.0000E+00(≈)	0.0000E+00/0.0000E+00	0.0000E+00/0.0000E+00(≈)	0.0000E+00/0.0000E+00	3.7896E−14/1.4677E−14(≈)	3.7896E−14/1.3531E−14
f_{a_3}	0.0000E+00/0.0000E+00(≈)	0.0000E+00/0.0000E+00	0.0000E+00/0.0000E+00(≈)	0.0000E+00/0.0000E+00	5.4614E−14/1.1144E−14(≈)	5.6843E−14/0.0000E+00
f_{a_4}	2.7279E+01/1.4447E+01(≈)	2.8131E+01/1.3621E+01	1.1146E−14/2.2793E−14(≈)	1.0031E−14/2.1885E−14	**1.8384E−01/2.0714E−01**(>)	2.1784E+00/1.3703E+01
f_{a_5}	1.2533E+01/9.7519E+00(≈)	8.2224E+00/9.9228E+00	2.0024E+01/6.9187E−02(≈)	2.0013E+01/3.5268E−02	2.0050E+01/5.4570E−02(≈)	2.0052E+01/7.1318E−02
f_{a_6}	0.0000E+00/0.0000E+00(≈)	0.0000E+00/0.0000E+00	0.0000E+00/0.0000E+00(≈)	0.0000E+00/0.0000E+00	2.7115E−04/6.0022E−04(≈)	1.7885E−02/1.2522E−01
f_{a_7}	1.4502E−04/1.0357E−03(≈)	2.2069E−13/8.6175E−13	0.0000E+00/0.0000E+00(≈)	0.0000E+00/0.0000E+00	9.8083E−14/3.9511E−14(≈)	9.5854E−14/4.1756E−14
f_{a_8}	1.7833E−14/4.1756E−14(≈)	2.4521E−14/4.7224E−14	**3.5221E−13/6.1243E−14**(>)	6.2305E−12/5.2658E−12	**2.9205E−07/1.4955E−07**(>)	2.3721E−05/1.1695E−05
f_{a_9}	1.5022E+00/8.5294E−01(≈)	1.4836E+00/7.0060E−01	1.6368E+01/6.4917E+00(<)	**1.2501E+01/4.5896E+00**	3.1550E+01/9.4578E+00(<)	**2.6018E+01/1.0122E+01**
$f_{a_{10}}$	4.8984E−03/1.6958E−02(≈)	1.2246E−03/8.7454E−03	2.5114E+00/1.3811E+00(≈)	2.7137E+00/1.7597E+00	4.3202E+01/1.0623E+01(≈)	4.1202E+01/8.5372E+00
$f_{a_{11}}$	4.0722E+01/6.4948E+01(≈)	3.2735E+01/5.2909E+01	1.3929E+03/5.4020E+02(<)	**1.1542E+03/2.6700E+02**	3.8728E+03/9.7198E+02(<)	**3.4504E+03/9.0005E+02**
$f_{a_{12}}$	3.8289E−02/1.7045E−02(≈)	3.9648E−02/1.6622E−02	1.1607E−01/2.2330E−02(≈)	1.2275E−01/2.3823E−02	1.5437E−01/2.8485E−02(≈)	1.5268E−01/2.6091E−02
$f_{a_{13}}$	3.3259E−02/6.7430E−03(≈)	3.4598E−02/9.2049E−03	1.0676E−01/1.7862E−02(≈)	1.1176E−01/1.7662E−02	1.9982E−01/2.5308E−02(<)	**1.8985E−01/2.4267E−02**
$f_{a_{14}}$	1.2162E−01/3.4420E−02(<)	**9.8220E−02/3.3909E−02**	1.8306E−01/2.0910E−02(≈)	1.8100E−01/2.1144E−02	2.5715E−01/1.9566E−02(≈)	2.5537E−01/1.8800E−02
$f_{a_{15}}$	3.8952E−01/9.1719E−02(≈)	3.8008E−01/7.0953E−02	2.6675E+00/7.0386E−01(≈)	2.4585E+00/6.3445E−01	5.1357E+00/1.0914E+00(≈)	5.2158E+00/1.0622E+00
$f_{a_{16}}$	1.0713E+00/3.4012E−01(≈)	1.1767E+00/3.9300E−01	8.7694E+00/5.5980E−01(≈)	8.6657E+00/5.0740E−01	1.6544E+01/5.9297E−01(≈)	1.6568E+01/5.0915E−01
$f_{a_{17}}$	6.3680E+00/5.0737E+00(<)	**4.5248E+00/4.0045E+00**	8.1994E+01/3.6986E+01(≈)	8.0892E+01/3.0385E+01	4.7920E+02/2.7569E+02(≈)	4.7525E+02/2.2806E+02
$f_{a_{18}}$	8.3506E−02/6.6444E−02(≈)	9.6501E−02/8.4223E−02	3.5725E+00/1.5185E+00(≈)	3.1368E+00/1.7736E+00	1.9064E+01/8.4165E+00(≈)	1.7749E+01/6.8338E+00
$f_{a_{19}}$	**2.6974E−02/2.0087E−02**(>)	3.8840E−02/2.5674E−02	2.4777E+00/4.4236E−01(≈)	2.4730E+00/5.6319E−01	**8.2083E+00/1.6369E+00**(>)	8.9199E+00/1.0915E+00
$f_{a_{20}}$	2.1026E−01/2.0586E−01(≈)	1.6841E−01/1.7021E−01	3.6774E+00/1.3794E+00(<)	**2.8310E+00/1.1558E+00**	9.8110E+00/2.2072E+00(≈)	9.0221E+00/2.5657E+00
$f_{a_{21}}$	2.5397E−01/1.8860E−01(≈)	2.6442E−01/1.1189E−01	3.0826E+01/5.2964E+01(≈)	2.7387E+01/4.0847E+01	3.4599E+02/1.2933E+02(≈)	3.1801E+02/8.4134E+01
$f_{a_{22}}$	**2.0940E−01/1.0413E−01**(>)	3.0802E−01/1.2816E−01	1.0949E+02/5.1699E+01(≈)	1.1117E+02/5.1117E+01	1.7825E+02/7.1588E+01(≈)	1.6599E+02/4.6424E+01
$f_{a_{23}}$	3.2946E+02/0.0000E+00(<)	**2.0000E+02/0.0000E+00**	3.1524E+02/0.0000E+00(<)	**2.0000E+02/0.0000E+00**	3.4400E+02/1.8402E−13(<)	**2.0000E+02/0.0000E+00**
$f_{a_{24}}$	1.0526E+02/3.0088E+00(≈)	1.0467E+02/3.1226E+00	2.2314E+02/6.5772E−01(<)	**2.0733E+02/1.0950E+01**	2.7154E+02/1.8368E+00(<)	**2.0000E+02/0.0000E+00**
$f_{a_{25}}$	1.2053E+02/1.4499E+01(≈)	1.1924E+02/1.0904E+01	2.0260E+02/5.7233E−02(<)	**2.0000E+02/0.0000E+00**	2.0548E+02/3.3590E−01(<)	**2.0000E+02/0.0000E+00**
$f_{a_{26}}$	1.0004E+02/8.8126E−03(≈)	1.0003E+02/1.0131E−02	1.0011E+02/1.3978E−02(≈)	1.0011E+02/1.6604E−02	1.4912E+02/5.0387E+01(≈)	1.5498E+02/5.0169E+01
$f_{a_{27}}$	5.4156E+01/1.2497E+02(≈)	2.8728E+01/6.8994E+01	3.0000E+02/0.0000E+00(<)	**2.0000E+02/0.0000E+00**	3.0422E+02/1.4912E+01(<)	**2.0000E+02/0.0000E+00**
$f_{a_{28}}$	3.8530E+02/4.5568E+01(<)	**2.2973E+02/6.6814E+01**	8.4788E+02/1.9907E+01(<)	**2.0000E+02/0.0000E+00**	1.2821E+03/6.9431E+01(<)	**2.0000E+02/0.0000E+00**
$f_{a_{29}}$	2.2175E+02/3.4582E−02(<)	**2.0000E+02/0.0000E+00**	1.4266E+02/8.5954E+01(≈)	1.3357E+02/2.6669E+01	4.7459E+02/1.2949E+02(<)	**2.0000E+02/0.0000E+00**
$f_{a_{30}}$	4.7582E+02/1.6464E+01(<)	**2.6408E+02/1.2380E+02**	4.1670E+02/3.8266E+01(<)	**2.0000E+02/0.0000E+00**	9.1437E+03/3.6549E+02(<)	**2.0000E+02/0.0000E+00**
>/≈/<	2/22/6	–/–/–	1/20/9	–/–/–	3/17/10	–/–/–

Table 11
Comparison results of the coefficients of scaling factor F for different difference vectors under the benchmark f_{b_1}–$f_{b_{30}}$ of CEC2017 test suite on 30D optimization. The overall performance (>, =or<) of each algorithm is measured under Wilcoxon's signed rank test with the significant level $\alpha = 0.05$ in comparison with the proposed ADEDMR. The bottom line summarized the overall performance on the 30 benchmarks.

CEC2014	10D		30D		50D	
Algorithm	ADEDMR[1]	ADEDMR	ADEDMR[1]	ADEDMR	ADEDMR[1]	ADEDMR
f_{b_1}	0.0000E+00/0.0000E+00(≈)	0.0000E+00/0.0000E+00	2.7864E−16/1.9899E−15(≈)	0.0000E+00/0.0000E+00	2.3685E−14/7.8696E−15(<)	**1.4768E−14/2.7859E−15**
f_{b_2}	0.0000E+00/0.0000E+00(≈)	0.0000E+00/0.0000E+00	0.0000E+00/0.0000E+00(≈)	0.0000E+00/0.0000E+00	2.3963E−14/1.8307E−14(<)	**5.0156E−15/1.0943E−14**
f_{b_3}	0.0000E+00/0.0000E+00(≈)	0.0000E+00/0.0000E+00	1.0031E−14/2.1885E−14(<)	**0.0000E+00/0.0000E+00**	1.6384E−13/5.0392E−14(<)	**7.3562E−14/3.2741E−14**
f_{b_4}	0.0000E+00/0.0000E+00(≈)	0.0000E+00/0.0000E+00	**4.8011E+01/2.3480E+01**(>)	5.7958E+01/8.4442E+00	7.2728E+01/4.7612E+01(<)	**5.3659E+01/3.9345E+01**
f_{b_5}	2.3431E+00/1.5128E+00(≈)	1.9519E+00/1.0133E+00	1.3988E+01/5.1868E+00(<)	**1.0388E+01/4.1155E+00**	2.4972E+01/7.9552E+00(≈)	2.6478E+01/1.0830E+01
f_{b_6}	2.0062E−14/4.3771E−14(≈)	2.6750E−14/4.8704E−14	1.1369E−13/0.0000E+00(≈)	1.1369E−13/0.0000E+00	2.2934E−08/7.5420E−08(<)	**9.4019E−10/6.7133E−09**
f_{b_7}	1.1898E+01/7.5960E−01(≈)	1.1773E+01/5.6448E−01	3.6583E+01/3.1034E+00(≈)	3.7143E+01/3.2028E+00	6.5514E+01/6.5074E+00(≈)	6.3689E+01/5.4302E+00
f_{b_8}	2.1851E+00/1.3932E+00(<)	**1.5022E+00/8.2940E−01**	1.1266E+01/4.9168E+00(≈)	1.0342E+01/4.4497E+00	2.7322E+01/9.1642E+00(≈)	2.3944E+01/8.0188E+00
f_{b_9}	0.0000E+00/0.0000E+00(≈)	0.0000E+00/0.0000E+00	0.0000E+00/0.0000E+00(≈)	0.0000E+00/0.0000E+00	4.2354E−14/5.5513E−14(<)	**0.0000E+00/0.0000E+00**
$f_{b_{10}}$	9.0769E+01/1.0575E+02(≈)	4.9169E+01/7.9379E+01	1.5434E+03/3.1424E+02(≈)	1.5108E+03/3.8999E+02	3.3027E+03/7.9835E+02(<)	**2.9811E+03/4.4109E+02**
$f_{b_{11}}$	4.4769E−02/3.1971E−01(≈)	4.8681E−02/2.5434E−01	5.2354E+00/8.5034E+00(<)	**3.7007E+00/8.3375E+00**	3.4921E+01/5.0856E+00(<)	**2.4973E+01/2.6976E+00**
$f_{b_{12}}$	2.5886E+00/1.6793E+01(≈)	2.6936E−01/1.6804E−01	3.2304E+02/1.8541E+02(<)	**1.0099E+02/9.0391E+01**	1.8178E+03/4.9013E+02(<)	**1.1829E+03/3.0401E+02**
$f_{b_{13}}$	5.5730E−01/1.4119E+00(≈)	7.2786E−01/1.7182E+00	1.3289E+01/5.7372E+00(<)	**1.1150E+01/6.6924E+00**	3.1608E+01/2.3895E+01(<)	**1.2624E+01/1.4463E+01**
$f_{b_{14}}$	2.7313E−01/4.4844E−01(≈)	2.9603E−01/5.3583E−01	2.1254E+01/5.4930E+00(≈)	2.1003E+01/5.4064E+00	2.9657E+01/3.1819E+00(<)	**2.7474E+01/2.5695E+00**
$f_{b_{15}}$	2.3153E−01/2.1493E−01(<)	**1.4891E−01/1.8979E−01**	2.2799E+00/1.0174E+00(<)	**8.2174E−01/5.6753E−01**	2.6885E+01/3.6697E+00(<)	**1.9617E+01/1.6646E+00**
$f_{b_{16}}$	4.4515E−01/2.2769E−01(≈)	3.8756E−01/2.3994E−01	1.2386E+02/1.0095E+02(≈)	1.3205E+02/1.0747E+02	3.7614E+02/1.1380E+02(≈)	3.5106E+02/1.2762E+02
$f_{b_{17}}$	2.3968E−01/4.3762E−01(≈)	3.4492E−01/6.5451E−01	2.5794E+01/9.3064E+00(≈)	2.4991E+01/7.8010E+00	2.7556E+02/9.2412E+01(≈)	3.0386E+02/1.0212E+02
$f_{b_{18}}$	2.5555E−01/2.1784E−01(<)	**1.4864E−01/2.2531E−01**	2.1073E+01/6.3304E−01(<)	**2.0696E+01/3.6939E−01**	2.4819E+01/2.5753E+00(<)	**2.1601E+01/1.0704E+00**
$f_{b_{19}}$	1.7050E−02/1.5486E−02(<)	**1.2750E−02/2.2777E−02**	3.8064E+00/1.2517E+00(≈)	3.4732E+00/1.2824E+00	1.6001E+01/2.7783E+00(<)	**1.1515E+01/2.4624E+00**
$f_{b_{20}}$	0.0000E+00/0.0000E+00(≈)	1.2242E−02/6.1198E−02	2.9262E+01/2.5653E+01(≈)	3.7382E+01/3.5003E+01	1.0416E+02/7.0718E+01(≈)	1.0400E+02/7.2738E+01
$f_{b_{21}}$	1.4481E+02/5.1972E+01(≈)	1.4031E+02/5.0693E+01	2.1104E+02/4.3917E+00(<)	**2.0945E+02/5.5556E+00**	2.2882E+02/9.9318E+00(<)	**2.2397E+02/8.8790E+00**
$f_{b_{22}}$	1.0001E+02/6.8312E−02(≈)	1.0001E+02/6.2341E−02	1.0000E+02/1.4352E−14(≈)	1.0000E+02/1.4352E−14	2.5673E+02/7.8362E+02(≈)	1.0004E+02/2.8524E−01
$f_{b_{23}}$	3.0119E+02/1.8846E+00(≈)	3.0064E+02/1.2922E+00	3.4933E+02/5.8292E+00(≈)	3.5001E+02/6.3984E+00	4.3902E+02/1.4239E+01(≈)	4.3769E+02/1.5941E+01
$f_{b_{24}}$	2.4705E+02/1.0977E+02(≈)	2.9958E+02/7.3829E+01	4.2474E+02/4.5601E+00(≈)	4.2677E+02/4.8010E+00	4.9839E+02/5.2710E+00(≈)	4.9907E+02/6.1659E+00
$f_{b_{25}}$	4.1300E+02/2.1671E+01(≈)	4.1037E+02/2.0493E+01	3.8673E+02/2.2327E−02(<)	**3.8670E+02/1.3603E−02**	5.1604E+02/2.3135E+01(<)	**4.8787E+02/1.5879E+01**
$f_{b_{26}}$	3.0000E+02/0.0000E+00(≈)	3.0000E+02/0.0000E+00	9.1462E+02/5.9685E+01(≈)	9.0958E+02/6.3085E+01	1.1259E+03/1.0115E+02(<)	**1.0168E+03/8.1115E+01**
$f_{b_{27}}$	3.9271E+02/1.9862E+00(≈)	3.9257E+02/1.9283E+00	4.9684E+02/5.7489E+00(<)	**4.8918E+02/6.5468E+00**	5.2110E+02/1.0648E+01(<)	**5.0719E+02/7.9972E+00**
$f_{b_{28}}$	3.1113E+02/5.5625E+01(≈)	3.3338E+02/9.2329E+01	3.0629E+02/2.5421E+01(≈)	3.0670E+02/2.7084E+01	4.7322E+02/2.2478E+01(<)	**4.6172E+02/1.1607E+01**
$f_{b_{29}}$	2.3184E+02/3.2129E+00(≈)	2.3192E+02/3.5060E+00	4.3659E+02/1.3917E+01(≈)	4.3217E+02/1.1629E+01	3.9463E+02/2.7951E+01(<)	**3.7413E+02/3.6971E+01**
$f_{b_{30}}$	3.9451E+02/2.5690E−02(≈)	3.9450E+02/5.6401E−03	1.9866E+03/2.6880E+01(<)	**1.9796E+03/1.1510E+01**	6.1005E+05/3.4092E+04(<)	**5.9571E+05/2.5540E+04**
>/≈/<	0/26/4	–/–/–	1/18/11	–/–/–	0/9/21	–/–/–

Table 12
Summary of the comparison results between ADEDMR and the other contrasted algorithms under the CEC2014, CEC2017 and CEC2022 benchmarks optimization.

A given algorithm versus ADEDMR											
Test suit:	CEC2014				CEC2017				CEC2022		
>/≈/<	D=10	D=30	D=50	∑	D=10	D=30	D=50	∑	D=10	D=20	∑
LSHADE	6/13/11	5/6/19	6/2/2	17/21/52	2/16/12	5/10/15	5/3/22	12/29/49	0/10/2	1/8/3	1/18/5
HARD-DE	5/12/13	2/8/20	5/3/22	12/23/55	2/19/9	1/8/21	1/4/25	4/31/55	1/7/4	1/5/6	2/12/10
NDE	3/8/19	4/3/23	2/4/24	9/15/66	7/8/15	2/2/26	0/3/27	9/13/68	1/7/4	1/6/5	2/13/9
ISDE	1/8/21	2/4/24	2/4/24	5/16/69	3/11/16	0/4/26	0/0/30	3/15/72	1/8/3	1/5/6	2/13/9
PaDE	4/18/8	5/10/15	7/5/18	16/33/41	4/10/16	5/11/14	5/2/23	14/23/53	0/9/3	0/10/2	0/19/5
CS-DE	3/16/11	7/5/18	5/8/17	15/29/46	0/18/12	1/10/19	0/7/23	1/35/54	1/7/4	1/7/4	2/14/8

Table 13
ADEDMR obtains the best or tie best in comparison with a given algorithm on 30D optimization under $f_{a_1} - f_{a_{30}}$ of our test suite.

Algorithm	Benchmarks on which ADEDMR wins or obtains similar performance
LSHADE	$f_{a_1} - f_{a_4}$, f_{a_6}, f_{a_7}, $f_{a_9} - f_{a_{12}}$, $f_{a_{14}} - f_{a_{21}}$, $f_{a_{23}} - f_{a_{25}}$, $f_{a_{26}} - f_{a_{30}}$
HARD-DE	$f_{a_1} - f_{a_9}$, $f_{a_{11}} - f_{a_{25}}$, $f_{a_{27}} - f_{a_{30}}$
NDE	$f_{a_1} - f_{a_4}$, f_{a_6}, f_{a_7}, f_{a_9}, $f_{a_{11}}$, $f_{a_{13}} - f_{a_{30}}$
ISDE	$f_{a_1} - f_{a_7}$, $f_{a_9} - f_{a_{12}}$, $f_{a_{14}} - f_{a_{30}}$
PaDE	$f_{a_1} - f_{a_7}$, f_{a_9}, $f_{a_{11}}$, $f_{a_{12}}$, $f_{a_{14}} - f_{a_{21}}$, $f_{a_{23}} - f_{a_{25}}$, $f_{a_{27}} - f_{a_{30}}$
CS-DE	$f_{a_1} - f_{a_4}$, f_{a_6}, f_{a_7}, f_{a_9}, $f_{a_{11}}$, $f_{a_{14}} - f_{a_{21}}$, $f_{a_{23}} - f_{a_{25}}$, $f_{a_{27}} - f_{a_{30}}$

Table 14
A certain algorithm obtains the best or tier best performance on 30D optimization under $f_{a_1} - f_{a_{30}}$ of our test suite.

Algorithm	Benchmark on which a certain algorithm wins or obtains similar performance	Total
LSHADE	f_{a_2}, f_{a_3}, $f_{a_5} - f_{a_7}$, $f_{a_{13}}$, $f_{a_{22}}$, $f_{a_{26}}$	8
HARD-DE	f_{a_2}, f_{a_3}, f_{a_7}	3
NDE	f_{a_2}, f_{a_3}, f_{a_7}, $f_{a_{10}}$, $f_{a_{12}}$	5
ISDE	f_{a_7}, f_{a_8}, $f_{a_{16}}$	3
PaDE	f_{a_2}, f_{a_3}, f_{a_6}, f_{a_7}, f_{a_9}, $f_{a_{11}}$, $f_{a_{15}}$, $f_{a_{16}}$, $f_{a_{26}}$	9
CS-DE	f_{a_2}, f_{a_3}, f_{a_6}, f_{a_7}, $f_{a_{15}}$, $f_{a_{20}} - f_{a_{22}}$	12
ADEDMR	$f_{a_1} - f_{a_4}$, f_{a_6}, f_{a_7}, f_{a_9}, $f_{a_{11}}$, $f_{a_{14}}$, $f_{a_{17}} - f_{a_{21}}$, $f_{a_{23}} - f_{a_{30}}$	22

Table 15
The analysis of Wavelet Walk on CEC2014 30D optimization under benchmarks $f_{a_1}–f_{a_{30}}$ of our test suite. The overall performance (>, =or<) is measured under Wilcoxon's signed rank test with the significant level $\alpha = 0.05$ in comparison with the proposed ADEDMR.

CEC2014	10D		30D		50D	
Algorithm	ADEDMR[1]	ADEDMR	ADEDMR[1]	ADEDMR	ADEDMR[1]	ADEDMR
f_{a_1}	0.0000E+00/0.0000E+00(≈)	0.0000E+00/0.0000E+00	3.9010E−15/6.4049E−15(≈)	2.7864E−15/5.6983E−15	1.3861E−09/3.7658E−09(≈)	1.3644E−09/2.7526E−09
f_{a_2}	0.0000E+00/0.0000E+00(≈)	0.0000E+00/0.0000E+00	0.0000E+00/0.0000E+00(≈)	0.0000E+00/0.0000E+00	2.8979E−14/3.9798E−15(≈)	2.8422E−14/0.0000E+00
f_{a_3}	0.0000E+00/0.0000E+00(≈)	0.0000E+00/0.0000E+00	0.0000E+00/0.0000E+00(≈)	0.0000E+00/0.0000E+00	**3.1208E−14/2.8566E−14**(>)	4.4583E−14/2.3612E−14
f_{a_4}	2.2505E+01/1.6786E+01(≈)	2.2502E+01/1.6784E+01	4.4583E−15/1.5434E−14(≈)	1.1146E−15/7.9597E−15	1.9348E+00/1.3736E+01(≈)	1.9452E+00/1.3734E+01
f_{a_5}	1.3735E+01/9.3793E+00(≈)	1.4132E+01/9.2129E+00	2.0178E+01/9.4765E−02(≈)	2.0183E+01/8.5040E−02	2.0341E+01/1.2706E−01(≈)	2.0342E+01/9.0558E−02
f_{a_6}	0.0000E+00/0.0000E+00(≈)	0.0000E+00/0.0000E+00	0.0000E+00/0.0000E+00(≈)	0.0000E+00/0.0000E+00	0.0000E+00/0.0000E+00(≈)	5.5377E−07/3.9547E−06
f_{a_7}	2.4158E−03/4.6351E−03(≈)	3.9119E−03/8.0765E−03	0.0000E+00/0.0000E+00(≈)	0.0000E+00/0.0000E+00	6.6875E−15/2.7016E−14(≈)	8.9166E−15/3.0869E−14
f_{a_8}	0.0000E+00/0.0000E+00(≈)	0.0000E+00/0.0000E+00	2.2292E−13/1.2236E−13(≈)	3.5221E−13/5.2604E−13	**3.4108E−08/4.9444E−08**(>)	7.0251E−08/9.3304E−08
f_{a_9}	1.7172E+00/1.0728E+00(≈)	1.7931E+00/1.2253E+00	9.8382E+00/4.7225E+00(≈)	8.8634E+00/4.2692E+00	2.0134E+01/1.0647E+01(≈)	2.2192E+01/1.1619E+01
$f_{a_{10}}$	2.2043E−02/3.7103E−02(≈)	3.0615E−02/3.8240E−02	**2.8575E−03/8.3481E−03**(>)	2.4493E−03/7.9518E−03	7.3124E−03/8.7850E−03(≈)	1.0748E−02/1.1566E−02
$f_{a_{11}}$	5.5433E+01/9.2977E+01(≈)	4.0865E+01/8.4252E+01	1.2190E+03/3.5462E+02(≈)	1.2188E+03/3.2984E+02	3.1996E+03/3.9739E+02(≈)	3.2757E+03/4.0320E+02
$f_{a_{12}}$	6.2361E−02/3.7482E−02(≈)	6.4001E−02/3.2045E−02	1.8528E−01/5.8940E−02(≈)	1.8080E−01/4.6190E−02	2.3690E−01/4.5958E−02(≈)	2.4387E−01/6.0673E−02
$f_{a_{13}}$	5.7496E−02/1.6930E−02(<)	**5.2020E−02/1.6516E−02**	1.5194E−01/2.7015E−02(≈)	1.5589E−01/2.4722E−02	2.3476E−01/2.9609E−02(≈)	2.3594E−01/2.8194E−02
$f_{a_{14}}$	8.3176E−02/3.2190E−02(≈)	9.1202E−02/3.6244E−02	1.8738E−01/1.7152E−02(≈)	1.8729E−01/2.0456E−02	2.3555E−01/2.0392E−02(≈)	2.3029E−01/1.7568E−02
$f_{a_{15}}$	3.6296E−01/6.5689E−02(≈)	3.7115E−01/7.2697E−02	2.2045E+00/5.1899E−01(≈)	2.2360E+00/3.6399E−01	5.1207E+00/6.8597E−01(≈)	5.0047E+00/7.4545E−01
$f_{a_{16}}$	9.1234E−01/3.7578E−01(≈)	9.1768E−01/3.7221E−01	8.6568E+00/5.6590E−01(≈)	8.4712E+00/5.1129E−01	1.6644E+01/4.3909E−01(≈)	1.6605E+01/4.8650E−01
$f_{a_{17}}$	2.3817E+00/3.4549E+00(≈)	1.4838E+00/1.9092E+00	5.8936E+01/2.4271E+01(≈)	5.9657E+01/1.4654E+01	2.3702E+02/1.5738E+02(≈)	2.9856E+02/2.1377E+02
$f_{a_{18}}$	5.6159E−02/6.4581E−02(≈)	4.9698E−02/5.1362E−02	2.4791E+00/1.4603E+00(≈)	2.3046E+00/1.3040E+00	6.1395E+00/2.3053E+00(≈)	6.2531E+00/2.0430E+00
$f_{a_{19}}$	4.7607E−02/5.0825E−02(≈)	3.9460E−02/3.3344E−02	1.8190E+00/5.1512E−01(≈)	1.7758E+00/5.5869E−01	7.9352E+00/1.2404E+00(≈)	7.9747E+00/1.5233E+00
$f_{a_{20}}$	2.1610E−01/1.8239E−01(≈)	2.4379E−01/1.9360E−01	2.0467E+00/8.2310E−01(≈)	2.1475E+00/9.4406E−01	5.2829E+00/1.9358E+00(≈)	5.3229E+00/2.1712E+00
$f_{a_{21}}$	2.9692E−01/2.6875E−01(≈)	3.2607E−01/2.5924E−01	9.3141E+00/4.8793E+00(≈)	1.0928E+01/2.1014E+01	2.5665E+02/7.1980E+01(≈)	2.6066E+02/7.5228E+01
$f_{a_{22}}$	1.2297E−01/6.2379E−02(≈)	1.2703E−01/4.8132E−02	8.3377E+01/5.9303E+01(≈)	9.8007E+01/5.7007E+01	1.5789E+02/6.1295E+01(≈)	1.6249E+02/7.5219E+01
$f_{a_{23}}$	3.2946E+02/0.0000E+00(<)	**2.0000E+02/0.0000E+00**	3.1524E+02/9.0949E−14(<)	**2.0000E+02/0.0000E+00**	3.4400E+02/2.0017E−13(<)	**2.0000E+02/0.0000E+00**
$f_{a_{24}}$	1.0521E+02/3.2010E+00(≈)	1.0605E+02/2.9750E+00	2.1579E+02/9.8163E+00(<)	**2.0085E+02/4.2578E+00**	2.6946E+02/2.1738E+00(<)	**2.0000E+02/0.0000E+00**
$f_{a_{25}}$	1.1881E+02/2.1825E+01(≈)	1.1978E+02/2.4756E+01	2.0258E+02/2.7798E−02(<)	**2.0036E+02/8.9944E−01**	2.0510E+02/2.4540E−01(<)	**2.0000E+02/0.0000E+00**
$f_{a_{26}}$	1.0005E+02/2.1322E−02(≈)	1.0005E+02/1.7856E−02	1.0016E+02/2.2081E−02(≈)	1.0015E+02/2.1741E−02	1.0024E+02/3.0024E−02(≈)	1.0024E+02/3.0053E−02
$f_{a_{27}}$	4.4430E+01/1.0950E+02(≈)	1.3161E+01/4.7176E+01	3.0000E+02/0.0000E+00(<)	**2.0000E+02/0.0000E+00**	3.0093E+02/6.6166E+00(<)	**2.0000E+02/0.0000E+00**
$f_{a_{28}}$	3.7364E+02/4.4185E+01(<)	**2.6888E+02/9.1401E+01**	8.1594E+02/2.1778E+01(<)	**2.0000E+02/0.0000E+00**	1.1352E+03/3.7162E+01(<)	**2.0000E+02/0.0000E+00**
$f_{a_{29}}$	2.2176E+02/4.0902E−03(<)	**2.0043E+02/3.0481E+00**	1.8964E+02/1.9372E+02(≈)	1.2501E+02/2.8523E+01	4.1029E+02/6.4713E+01(<)	**2.0000E+02/0.0000E+00**
$f_{a_{30}}$	4.5884E+02/2.2367E+01(<)	**2.6595E+02/1.1034E+02**	4.0203E+02/2.3959E+01(<)	**2.0000E+02/0.0000E+00**	8.3940E+03/7.0821E+02(<)	**1.1595E+03/2.6561E+03**
>/≈/<	0/25/5	–/–/–	1/23/6	–/–/–	2/21/7	–/–/–

Table 16
Four different parameter-controlled versions of ADEDMR.

Versions	CR	F
ADEDMR$_1$	Laplace (μ_{CR},0.1)	Laplace (μ_F,0.2)
ADEDMR$_2$	Gaussian (μ_{CR},0.1)	Cauchy (μ_F,0.1)
ADEDMR$_3$	Laplace (μ_{CR},0.1)	Gaussian (μ_F,0.1)
ADEDMR$_4$	Gaussian (μ_{CR},0.1)	Laplace (μ_F,0.2)
ADEDMR	Laplace (μ_{CR},0.1)	Cauchy (μ_F,0.1)

may be that each time the probability distribution random number is generated for F, the original good F in the historical archive μ_F has become bad, resulting in the search in a hopeless direction.

4.6. Evaluate the time complexity of the ADEDMR

This section presents the time complexity evaluation of the newly proposed ADEDMR. What can be seen is that the proposed ADEDMR algorithm merely consume more time compared to LSHADE and PaDE, mainly attributed to its two components, one is the extra difference operation in the mutation strategy ($X_{r_3,G}-\widehat{X}_{r_4,G}$) (as in Eq. (10)), which provides depth information for evolution, and a restart mechanism with diversity evaluation (pseudocode shown in Algorithms 1 and Algorithms 2. In order to better compare the time complexity of each algorithm, we can simply rank the time complexity of these algorithms by qualitative analysis.

Beside the qualitative analysis, we also conducted quantitative analysis according to the evaluation criteria of the CEC2014 competition (Liang et al., 2013a), and the time consumption comparison is shown in Table 18. The time consumption of standard program running in the recommendation of the CEC2014 competition is recorded as T_0, the time consumption of the 200,000 function evaluation of the 30D optimization of the benchmark $f_{a_{18}}$ by the CEC2014 test suite is recorded as T_1, and overall cost of optimizing $f_{a_{18}}$ by an algorithm is recorded as T_2. We run 11 times to get the average T_0, T_1 and T_2, then collect T_2, T_1 and T_0 for complexity evaluation. Table 18 illustrates the time complexity comparison between these seven algorithms, we can see that the time consumption of the proposed ADEDMR algorithm is less than that of HARD-DE, NDE, ISDE, and CS-DE, and only consumes more than LSHADE and PaDE.

4.7. Photovoltaic model parameter extraction

Photovoltaic power generation has the advantages of fast response to grid frequency regulation, no noise, and easy installation (Parida et al., 2011). With these advantages, photovoltaic power generation has become one of the fastest-growing emerging industries in renewable energy. Accurate simulation, evaluation and control of photovoltaic power generation systems depend largely on the accuracy of photovoltaic models and their parameters. Currently, there are two most commonly used photovoltaic models, the single diode model (SDM) and the double diode model (DDM). Similar to Li et al. (2020), this study takes root mean square error (RMSE) as the objective function and transforms it into a minimum optimization problem:

$$\mathrm{RMSE}(X)=\sqrt{\frac{1}{N}\sum_{k=1}^{N}f_k(V,I,X)^2} \tag{26}$$

Where X is the solution vector composed of unknown parameters, and N is the number of experimental data. I and V are the solar cell output current and output voltage, respectively. The equivalent circuit of the double diode model is given in 7, which consists of a current source, two diodes and two resistors. The output current I can be calculated as follows Eq. (27):

$$I=I_{ph}-I_{d_1}-I_{d_2}-I_{sh} \tag{27}$$

where I_{d_1} and I_{d_2} represent the first and second diode currents respectively, which can be expressed as:

$$I_{d_1}=I_o[\exp(\frac{V+IR_s}{a_1V_t})-1] \tag{28}$$

$$I_{d_2}=I_o[\exp(\frac{V+IR_s}{a_2V_t})-1] \tag{29}$$

I_{sh} denote the shunt resistor current, and it is calculated as below:

$$I_{sh}=\frac{V+IR_s}{R_{sh}} \tag{30}$$

where I_o, a_1, a_2, R_s, R_{sh} represent diode reverse saturation current, the first and second diode ideality factor, series resistance and shunt resistance, respectively. V is the battery output voltage and V_t is the junction thermal voltage, defined as follows:

$$V_t=\frac{k\cdot T}{q} \tag{31}$$

where k denotes the Boltzmann constant (1.3806503×10^{-23} J/K), T is the temperature of junction in Kelvin, and q denotes the electron charge (1.60217646×10^{-19} C). Therefore, the error for each pair of experimental and simulated current data points is used as the objective function, where seven unknown parameters, including I_o, a_1, a_2, R_s, R_{sh}, I_{d_1}, I_{d_2}.

$$\begin{cases} f(V,I,X)=I_{ph}-I_{o_1}\cdot[\exp(\frac{V+IR_s}{a_1V_t})-1] \\ -I_{o_2}\cdot[\exp(\frac{V+IR_s}{a_2V_t})-1]-\frac{V+IR_s}{R_{sh}}-I \\ X=\{I_{ph},I_{o_1},I_{o_2},R_s,R_{sh},a_1,a_2\} \end{cases} \tag{32}$$

Due to the high cost of experimental equipment, the benchmark experimental current and voltage data of solar cells and solar modules come from Easwarakhanthan et al. (1986). To demonstrate the effectiveness of ADEDMR, six state-of-the-art algorithms, LSHADE, HARD-DE, NDE, ISDE, PaDE and CS-DE, are also used for comparison. The parameters of the above algorithms all adopt the default settings, such as Table 1. Table 19 shows the numerical results after 30 runs (10,000 function evaluations per run). It is obvious that ADEDMR obtains the best results. Therefore, ADEDMR is a more efficient algorithm to solve this problem.

4.8. Parameter estimation for frequency-modulated (FM) sound waves

Frequency modulation (FM) sound wave synthesis is used in the field of modern music systems and plays an important role. Optimizing the parameters of an FM synthesizer is a six-dimensional optimization problem, where the vector $X=\{k_1,\omega_1,k_2,\omega_2,k_3,\omega_3\}$ to be optimized is the estimated target sound and sounds similar to the target, as shown in Eqs. (33) and (34). This problem is a highly complex multimodal one having strong epistasis, with minimum value $f(X_{sol})=0$. This problem is solved in Sun and Wang (2015) using Particle Swarm Optimization (PSO). The estimated sound wave and the target sound wave are defined as follows:

$$y(t)=k_1\cdot\sin\left(\omega_1\cdot t\cdot\theta+k_2\cdot\sin\left(\omega_2\cdot t\cdot\theta+k_3\cdot\sin\left(\omega_3\cdot t\cdot\theta\right)\right)\right) \tag{33}$$

$$\begin{aligned} y_0(t)=&(1.0)\cdot\sin((5.0)\cdot t\cdot\theta-(1.5)\cdot\sin((4.8)\cdot t\cdot\theta) \\ &+((2.0)\cdot\sin((4.9)\cdot t\cdot\theta))) \end{aligned} \tag{34}$$

where $\theta=\frac{2\pi}{100}$, and all parameters are restricted in the range [−6.4, 6.35]. The population size of all algorithms is set to 30. The fitness function is the sum of the squared errors of the estimated wave and the target wave as follows:

$$f(X)=\sum_{t=0}^{100}\left(y(t)-y_0(t)\right)^2 \tag{35}$$

Table 17
The comparison results of different F controls on CEC2014 30D optimization under benchmarks f_{a_1}–$f_{a_{30}}$ of our test suite. The overall performance (>, =or<) is measured under Wilcoxon's signed rank test with the significant level $\alpha = 0.05$ in comparison with the proposed ADEDMR.

Algorithm	ADEDMR$_1$	ADEDMR$_2$	ADEDMR$_3$	ADEDMR$_4$	ADEDMR
f_{a_1}	3.6224E−15/6.2548E−15(≈)	3.0651E−15/5.9030E−15(≈)	2.5078E−15/5.4714E−15(≈)	3.9010E−15/6.4049E−15(≈)	2.7864E−15/5.6983E−15
f_{a_2}	0.0000E+00/0.0000E+00(≈)	0.0000E+00/0.0000E+00(≈)	0.0000E+00/0.0000E+00(≈)	0.0000E+00/0.0000E+00(≈)	0.0000E+00/0.0000E+00
f_{a_3}	0.0000E+00/0.0000E+00(≈)	0.0000E+00/0.0000E+00(≈)	0.0000E+00/0.0000E+00(≈)	0.0000E+00/0.0000E+00(≈)	0.0000E+00/0.0000E+00
f_{a_4}	4.4583E−15/1.5434E−14(≈)	2.2292E−15/1.1144E−14(≈)	5.5729E−15/1.7072E−14(≈)	5.5729E−15/1.7072E−14(≈)	1.1146E−15/7.9597E−15
f_{a_5}	2.0204E+01/8.3242E−02(≈)	2.0118E+01/1.3043E−01(>)	2.0294E+01/8.7868E−02(<)	2.0126E+01/1.2872E−01(>)	2.0183E+01/8.5040E−02
f_{a_6}	0.0000E+00/0.0000E+00(≈)	0.0000E+00/0.0000E+00(≈)	0.0000E+00/0.0000E+00(≈)	0.0000E+00/0.0000E+00(≈)	0.0000E+00/0.0000E+00
f_{a_7}	0.0000E+00/0.0000E+00(≈)	0.0000E+00/0.0000E+00(≈)	0.0000E+00/0.0000E+00(≈)	0.0000E+00/0.0000E+00(≈)	0.0000E+00/0.0000E+00
f_{a_8}	3.8119E−13/5.1840E−13(≈)	2.4298E−13/3.3108E−13(≈)	2.4958E−11/1.6180E−10(<)	2.5412E−13/6.3322E−13(>)	3.5221E−13/5.2604E−13
f_{a_9}	1.3412E+01/8.1210E+00(<)	1.2119E+01/6.7557E+00(<)	**6.6648E+00/2.0231E+00**(>)	1.4637E+01/6.2392E+00(<)	8.8634E+00/4.2692E+00
$f_{a_{10}}$	1.2247E−03/4.9474E−03(≈)	8.9809E−03/1.3969E−02(<)	2.8575E−03/7.2355E−03(≈)	5.7151E−03/1.1078E−02(<)	2.4493E−03/7.9518E−03
$f_{a_{11}}$	1.2585E+03/3.6048E+02(≈)	1.3679E+03/4.1784E+02(<)	1.2662E+03/2.7468E+02(≈)	1.3785E+03/4.2946E+02(<)	1.2188E+03/3.2984E+02
$f_{a_{12}}$	1.7199E−01/5.7740E−02(≈)	1.5771E−01/7.3037E−02(≈)	3.2964E−01/1.2483E−01(<)	**1.2777E−01/8.1996E−02**(>)	1.8080E−01/4.6190E−02
$f_{a_{13}}$	1.5831E−01/2.4352E−02(≈)	1.5337E−01/2.3338E−02(≈)	1.5190E−01/2.7397E−02(≈)	**1.4581E−01/2.6637E−02**(>)	1.5589E−01/2.4722E−02
$f_{a_{14}}$	1.9145E−01/2.5233E−02(≈)	1.9179E−01/2.8021E−02(≈)	2.0488E−01/2.0088E−02(<)	1.9989E−01/3.1732E−02(<)	1.8729E−01/2.0456E−02
$f_{a_{15}}$	2.2720E+00/4.8230E−01(≈)	2.1717E+00/4.0195E−01(≈)	2.1180E+00/2.5816E−01(≈)	2.5382E+00/5.9441E−01(<)	2.2360E+00/3.6399E−01
$f_{a_{16}}$	8.4287E+00/4.7274E−01(≈)	8.8253E+00/4.4627E−01(<)	8.5097E+00/5.9290E−01(≈)	8.5428E+00/6.3910E−01(≈)	8.4712E+00/5.1129E−01
$f_{a_{17}}$	6.7759E+01/2.4001E+01(≈)	5.7538E+01/1.4158E+01(≈)	8.6830E+01/4.0976E+01(<)	6.2942E+01/2.9114E+01(≈)	5.9657E+01/1.4654E+01
$f_{a_{18}}$	2.6144E+00/1.3549E+00(≈)	2.4194E+00/1.2969E+00(≈)	2.4513E+00/1.3647E+00(≈)	2.7959E+00/1.3849E+00(<)	2.3046E+00/1.3040E+00
$f_{a_{19}}$	2.0275E+00/5.1322E−01(<)	1.7669E+00/5.9126E−01(≈)	2.3029E+00/5.2840E−01(<)	1.8265E+00/4.5808E−01(≈)	1.7758E+00/5.5869E−01
$f_{a_{20}}$	2.3719E+00/1.2087E+00(≈)	2.0874E+00/7.4282E−01(≈)	2.2682E+00/1.0943E+00(≈)	2.4024E+00/1.0968E+00(≈)	2.1475E+00/9.4406E−01
$f_{a_{21}}$	1.4195E+01/2.7104E+01(≈)	1.4220E+01/2.5240E+01(≈)	3.5924E+01/4.9736E+01(<)	1.9093E+01/3.4685E+01(<)	1.0928E+01/2.1014E+01
$f_{a_{22}}$	8.8695E+01/5.9652E+01(≈)	9.3280E+01/5.7228E+01(≈)	1.1575E+02/4.8047E+01(≈)	9.6523E+01/5.7479E+01(≈)	9.8007E+01/5.7007E+01
$f_{a_{23}}$	2.0000E+02/0.0000E+00(≈)	2.0000E+02/0.0000E+00(≈)	2.0000E+02/0.0000E+00(≈)	2.0000E+02/0.0000E+00(≈)	2.0000E+02/0.0000E+00
$f_{a_{24}}$	2.0729E+02/1.0418E+01(<)	2.0041E+02/2.9477E+00(≈)	2.1835E+02/8.0042E+00(<)	2.0897E+02/1.0835E+01(<)	2.0085E+02/4.2578E+00
$f_{a_{25}}$	2.0021E+02/7.1503E−01(≈)	2.0026E+02/7.8152E−01(≈)	2.0005E+02/3.6200E−01(≈)	2.0020E+02/6.9934E−01(≈)	2.0036E+02/8.9944E−01
$f_{a_{26}}$	1.0016E+02/2.4292E−02(≈)	1.0015E+02/2.4557E−02(≈)	1.0015E+02/2.3866E−02(≈)	1.0015E+02/2.2397E−02(≈)	1.0015E+02/2.1741E−02
$f_{a_{27}}$	2.0000E+02/0.0000E+00(≈)	2.0000E+02/0.0000E+00(≈)	2.0000E+02/0.0000E+00(≈)	2.0000E+02/0.0000E+00(≈)	2.0000E+02/0.0000E+00
$f_{a_{28}}$	2.0000E+02/0.0000E+00(≈)	2.0000E+02/0.0000E+00(≈)	2.0000E+02/0.0000E+00(≈)	2.0000E+02/0.0000E+00(≈)	2.0000E+02/0.0000E+00
$f_{a_{29}}$	1.3827E+02/3.6950E+01(≈)	1.2520E+02/2.6932E+01(≈)	1.5608E+02/3.9237E+01(<)	1.3292E+02/3.4919E+01(≈)	1.2501E+02/2.8523E+01
$f_{a_{30}}$	2.0000E+02/0.0000E+00(≈)	2.0356E+02/2.5406E+01(≈)	2.0578E+02/4.1246E+01(≈)	2.0000E+02/0.0000E+00(≈)	2.0000E+02/0.0000E+00
>/≈/<	0/27/3	1/25/4	1/20/9	4/18/8	–/–/–

Table 18
Algorithm time complexity comparison on 30D optimization under CEC2014 benchmark 18.

Algorithms.	T_0	T_1	$\widehat{T}_2$	$\frac{\widehat{T}_2 - T_1}{T_0}$
LSHADE			0.5687	7.1141
PaDE			0.6920	9.1980
HARD-DE			1.5767	24.0419
ISDE	0.0596	0.1438	1.9530	37.3674
CS-DE			2.3709	37.3674
NDE			3.8691	62.5050
ADEDMR			1.1203	16.3842

To highlight the effectiveness of ADEDMR, six powerful algorithms LSHADE, HARD-DE, NDE, ISDE, PaDE and CS-DE are also used for comparison. Table 20 gives the numerical results after 30 runs (10,000 function evaluations per run). The results show that ADEDMR is competitive with the above six algorithms and is a promising algorithm to solve this problem.

5. Conclusion

To improve the search efficiency and convergence accuracy of the DE algorithm, and reduce the time complexity of solving large-scale complex problems, a hybrid mutation strategy based on depth information mutation strategy and adaptive parameter algorithm (ADEMDR) is designed to propose an improved DE algorithm in this paper. The new algorithm has three main contributions. The first one is to propose a deeply-informed mutation strategy. In the early stage of the search, the local individual information is used to accelerate the convergence, in the middle stage of the search, the depth information of the suboptimal solution after greedy selection is used to search for more individual information, and the search direction is adjusted in the later stage of the search to avoid the search stagnation. The second contribution is to propose a new parameter adaptation scheme. The Laplace distribution model adopted for CR can help the algorithm achieve a better balance between development and exploration. For F, an individual-level parameter Fl is set to control the generation of F segmentally, aiming at reducing the misleading generation of F. The third contribution is to propose an enhanced restart mechanism with wavelet walks. By assigning more appropriate search directions and parameters to each individual through this mechanism, the search needs of each individual can be effectively met and the diversity of the population can be maintained.

However, there is a certain difference between the optimal value and the average value, which indicates that the ADEDMR algorithm is not stable enough and needs to be further strengthened, and the optimization performance can still be improved. At the same time, the fixed ratio of ADEDMR may not be as effective as the adaptive strategy in selecting suboptimal solutions. In the future, we will further refine ADEDMR. Furthermore, we will test the efficiency of some components, such as additional DE variants, CMA-ES variants (Hansen et al., 2003; Knight and Lunacek, 2007), QUATRE variants (Meng and Pan, 2016b, 2018) for comparison.

CRediT authorship contribution statement

Quanbin Zhang: Methodology, Software, Writing – original draft. **Zhenyu Meng:** Conceptualization, Methodology, Software, Supervision, Writing – review & editing.

Declaration of competing 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.

Data availability

Data will be made available on request.

Acknowledgments

This work is supported by the Natural Science Foundation of Fujian Province with the Grant No. 2021J05227 & No. 2023J01347 and Scientific Research Startup Foundation of Fujian University of Technology (GY-Z19013).

Table 19
Comparison of ADEDMR with other algorithms on double diode model.

Algorithm	I_{ph}(A)	I_{o_1}(μA)	R_s(Ω)	R_{sh}(Ω)	a_1	I_{o_2}(μA)	a_2	RMSE
LSHADE	0.7605	0.3065	0.0354	56.8714	1.4819	0.5795	1.93089	1.4236E−03
HARD-DE	0.7611	0.2312	0.0329	55.2596	1.4571	0.4301	1.8166	1.0197E−03
NDE	0.7493	0.0367	55.4854	2.0000	0.2260	1.4510	1.4510	9.8248E−04
ISDE	0.7590	0.1702	0.0328	82.5860	1.6197	0.4889	1.5418	2.2284E−03
PaDE	0.7608	0.1489	0.0365	54.1770	1.6919	0.2506	1.4635	9.8548E−04
CS-DE	0.7608	0.2117	0.0365	54.6178	1.8163	0.2654	1.4660	9.8515E−04
ADEDMR	0.7608	2.1065	0.0365	54.2502	2.9011	0.3030	1.4752	**9.8112E−04**

Table 20
Comparison of ADEDMR with other algorithms on FM.

Algorithm	k_1	ω_1	k_2	ω_2	k_3	ω_3	$f(X)$
LSHADE	−1.0000	−5.0000	−1.4999	−4.8000	−2.0000	4.9000	7.0791E−07
HARD-DE	−0.9997	−5.0000	1.5004	4.8000	−2.0002	−4.9000	8.2265E−03
NDE	0.9999	5.0000	−1.5000	4.8000	2.0000	4.9000	**0**
ISDE	−0.9899	0.5.0050	1.5220	4.7991	2.0016	4.9004	0.0141
PaDE	−1.0000	−5.0000	−1.5000	−4.8000	−2.0000	4.9000	**0**
CS-DE	0.9994	5.0016	1.5032	−4.9799	1.9988	−4.9001	8.3440E−04
ADEDMR	1.0000	5.0000	1.50005	−4.8000	−2.0000	4.9000	**0**

References

Ali, M.Z., Awad, N.H., Suganthan, P.N., Reynolds, R.G., 2016. An adaptive multipopulation differential evolution with dynamic population reduction. IEEE Trans. Cybern. 47 (9), 2768–2779.

Awad, N.H., Ali, M.Z., Suganthan, P.N., 2017. Ensemble sinusoidal differential covariance matrix adaptation with euclidean neighborhood for solving CEC2017 benchmark problems. In: 2017 IEEE Congress on Evolutionary Computation. CEC, IEEE, pp. 372–379.

Bratton, D., Kennedy, J., 2007. Defining a standard for particle swarm optimization. In: 2007 IEEE Swarm Intelligence Symposium. IEEE, pp. 120–127.

Brest, J., Greiner, S., Boskovic, B., Mernik, M., Zumer, V., 2006. Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. IEEE Trans. Evol. Comput. 10 (6), 646–657.

Brest, J., Maučec, M.S., Bošković, B., 2020. Differential evolution algorithm for single objective bound-constrained optimization: Algorithm j2020. In: 2020 IEEE Congress on Evolutionary Computation. CEC, IEEE, pp. 1–8.

Chakraborty, A., Kar, A.K., 2017. Swarm intelligence: A review of algorithms. Nature-Inspir. Comput. Optim. 475–494.

Das, S., Abraham, A., Konar, A., 2007. Automatic clustering using an improved differential evolution algorithm. IEEE Trans. Syst. Man Cybern. 38 (1), 218–237.

Das, S., Mullick, S.S., Suganthan, P.N., 2016. Recent advances in differential evolution–an updated survey. Swarm Evol. Comput. 27, 1–30.

Das, S., Suganthan, P.N., 2010. Differential evolution: A survey of the state-of-the-art. IEEE Trans. Evol. Comput. 15 (1), 4–31.

Deng, W., Xu, J., Song, Y., Zhao, H., 2021. Differential evolution algorithm with wavelet basis function and optimal mutation strategy for complex optimization problem. Appl. Soft Comput. 100, 106724.

Deng, W., Xu, J., Zhao, H., 2019. An improved ant colony optimization algorithm based on hybrid strategies for scheduling problem. IEEE Access 7, 20281–20292.

Dorigo, M., Birattari, M., Stutzle, T., 2006. Ant colony optimization. IEEE Comput. Intell. Mag. 1 (4), 28–39.

Easwarakhanthan, T., Bottin, J., Bouhouch, I., Boutrit, C., 1986. Nonlinear minimization algorithm for determining the solar cell parameters with microcomputers. Int. J. Solar Energy 4 (1), 1–12.

Hansen, N., Müller, S.D., Koumoutsakos, P., 2003. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). Evol. Comput. 11 (1), 1–18.

Holland, J.H., 1992. Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. MIT Press.

Hu, Y., Sim, C.-K., Yang, X., 2015. A subgradient method based on gradient sampling for solving convex optimization problems. Numer. Funct. Anal. Optim. 36 (12), 1559–1584.

Hu, Y., Yu, C.K.W., Li, C., 2016. Stochastic subgradient method for quasi-convex optimization problems. J. Nonlinear Convex Anal. 17 (4), 711–724.

Jeong, Y.-W., Park, J.-B., Jang, S.-H., Lee, K.Y., 2010. A new quantum-inspired binary PSO: application to unit commitment problems for power systems. IEEE Trans. Power Syst. 25 (3), 1486–1495.

Knight, J.N., Lunacek, M., 2007. Reducing the space-time complexity of the CMA-ES. In: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation. pp. 658–665.

Li, S., Gu, Q., Gong, W., Ning, B., 2020. An enhanced adaptive differential evolution algorithm for parameter extraction of photovoltaic models. Energy Convers. Manage. 205, 112443.

Li, Y., Han, T., Tang, S., Huang, C., Zhou, H., Wang, Y., 2023. An improved differential evolution by hybridizing with estimation-of-distribution algorithm. Inform. Sci. 619, 439–456.

Li, Y.-L., Zhan, Z.-H., Gong, Y.-J., Chen, W.-N., Zhang, J., Li, Y., 2014. Differential evolution with an evolution path: A DEEP evolutionary algorithm. IEEE Trans. Cybern. 45 (9), 1798–1810.

Liang, J.J., Qu, B.Y., Suganthan, P.N., 2013a. Problem Definitions and Evaluation Criteria for the CEC 2014 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization, Vol. 635. Technical Report, Nanyang Technological University, Singapore, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China, p. 490.

Liang, J.J., Qu, B., Suganthan, P.N., Hernández-Díaz, A.G., 2013b. Problem Definitions and Evaluation Criteria for the CEC 2013 Special Session on Real-Parameter Optimization, Vol. 201212, No. 34. Technical Report, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Nanyang Technological University, Singapore, pp. 281–295.

Liao, Z., Mi, X., Pang, Q., Sun, Y., 2023. History archive assisted niching differential evolution with variable neighborhood for multimodal optimization. Swarm Evol. Comput. 76, 101206.

Meng, Z., 2023. Dimension improvements based adaptation of control parameters in Differential Evolution: A fitness-value-independent approach. Expert Syst. Appl. 223, 119848.

Meng, Z., Chen, Y., 2023. Differential Evolution with exponential crossover can be also competitive on numerical optimization. Appl. Soft Comput. 146, 110750.

Meng, Z., Chen, Y., Li, X., Yang, C., Zhong, Y., 2020. Enhancing QUasi-Affine TRansformation evolution (QUATRE) with adaptation scheme on numerical optimization. Knowl.-Based Syst. 197, 105908.

Meng, Z., Pan, J.-S., 2016a. Monkey king evolution: a new memetic evolutionary algorithm and its application in vehicle fuel consumption optimization. Knowl.-Based Syst. 97, 144–157.

Meng, Z., Pan, J.-S., 2016b. Quasi-affine transformation evolutionary (QUATRE) algorithm: A parameter-reduced differential evolution algorithm for optimization problems. In: 2016 IEEE Congress on Evolutionary Computation. CEC, IEEE, pp. 4082–4089.

Meng, Z., Pan, J.-S., 2018. QUasi-Affine TRansformation evolution with external archive (QUATRE-EAR): an enhanced structure for differential evolution. Knowl.-Based Syst. 155, 35–53.

Meng, Z., Pan, J.-S., 2019. HARD-DE: Hierarchical archive based mutation strategy with depth information of evolution for the enhancement of differential evolution on numerical optimization. IEEE Access 7, 12832–12854.

Meng, Z., Pan, J.-S., Tseng, K.-K., 2019. PaDE: An enhanced differential evolution algorithm with novel control parameter adaptation schemes for numerical optimization. Knowl.-Based Syst. 168, 80–99.

Meng, Z., Yang, C., 2021. Hip-DE: Historical population based mutation strategy in differential evolution with parameter adaptive mechanism. Inform. Sci. 562, 44–77.

Meng, Z., Yang, C., 2022. Two-stage differential evolution with novel parameter control. Inform. Sci. 596, 321–342.

Meng, Z., Zhong, Y., Mao, G., Liang, Y., 2022. PSO-sono: a novel PSO variant for single-objective numerical optimization. Inform. Sci. 586, 176–191.

Meng, Z., Zhong, Y., Yang, C., 2021. CS-DE: Cooperative strategy based differential evolution with population diversity enhancement. Inform. Sci. 577, 663–696.

Mohamed, A.W., Hadi, A.A., Mohamed, A.K., 2021. Differential evolution mutations: taxonomy, comparison and convergence analysis. IEEE Access 9, 68629–68662.

Neri, F., Tirronen, V., 2010. Recent advances in differential evolution: a survey and experimental analysis. Artif. Intell. Rev. 33 (1), 61–106.

Osuna-Enciso, V., Cuevas, E., Castañeda, B.M., 2022. A diversity metric for population-based metaheuristic algorithms. Inform. Sci. 586, 192–208.

Parida, B., Iniyan, S., Goic, R., 2011. A review of solar photovoltaic technologies. Renew. Sustain. Energy Rev. 15 (3), 1625–1636.

Price, K., Storn, R.M., Lampinen, J.A., 2006. Differential Evolution: A Practical Approach to Global Optimization. Springer Science & Business Media.

Read, J., Martino, L., Luengo, D., 2014. Efficient monte carlo methods for multi-dimensional learning with classifier chains. Pattern Recognit. 47 (3), 1535–1546.

Salimi, H., 2015. Stochastic fractal search: a powerful metaheuristic algorithm. Knowl.-Based Syst. 75, 1–18.

Song, Y., Cai, X., Zhou, X., Zhang, B., Chen, H., Li, Y., Deng, W., Deng, W., 2023. Dynamic hybrid mechanism-based differential evolution algorithm and its application. Expert Syst. Appl. 213, 118834.

Storn, R., 1995. Differrential Evolution-A Simple and Efficient Adaptive Scheme for Global Optimization Over Continuous Spaces, Vol. 11. Technical Report, International Computer Science Institute.

Storn, R., Price, K., 1997. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. J. Global Optim. 11 (4), 341–359.

Sun, Y., Wang, Z., 2015. Asynchronous and stochastic dimension updating PSO and its application to parameter estimation for frequency modulated (FM) sound waves. In: 2015 IEEE International Conference on Progress in Informatics and Computing. PIC, IEEE, pp. 583–587.

Tanabe, R., Fukunaga, A., 2013. Success-history based parameter adaptation for differential evolution. In: 2013 IEEE Congress on Evolutionary Computation. IEEE, pp. 71–78.

Tanabe, R., Fukunaga, A.S., 2014. Improving the search performance of SHADE using linear population size reduction. In: 2014 IEEE Congress on Evolutionary Computation. CEC, IEEE, pp. 1658–1665.

Tian, M., Gao, X., 2019a. Differential evolution with neighborhood-based adaptive evolution mechanism for numerical optimization. Inform. Sci. 478, 422–448.

Tian, M., Gao, X., 2019b. An improved differential evolution with information intercrossing and sharing mechanism for numerical optimization. Swarm Evol. Comput. 50, 100341.

Tian, M., Gao, X., Yan, X., 2020. Performance-driven adaptive differential evolution with neighborhood topology for numerical optimization. Knowl.-Based Syst. 188, 105008.

Van Laarhoven, P.J., Aarts, E.H., 1987. Simulated annealing. In: Simulated Annealing: Theory and Applications. Springer, pp. 7–15.

Vikhar, P.A., 2016. Evolutionary algorithms: A critical review and its future prospects. In: 2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication. ICGTSPICC, IEEE, pp. 261–265.

Wilcoxon, F., 1992. Individual comparisons by ranking methods. In: Breakthroughs in Statistics. Springer, pp. 196–202.

Wu, G., Mallipeddi, R., Suganthan, P.N., 2017. Problem Definitions and Evaluation Criteria for the CEC 2017 Competition on Constrained Real-Parameter Optimization. Technical Report, National University of Defense Technology, Changsha, Hunan, PR China and Kyungpook National University, Daegu, South Korea and Nanyang Technological University, Singapore.

Wu, G., Shen, X., Li, H., Chen, H., Lin, A., Suganthan, P.N., 2018. Ensemble of differential evolution variants. Inform. Sci. 423, 172–186.

Yang, M., Li, C., Cai, Z., Guan, J., 2014. Differential evolution with auto-enhanced population diversity. IEEE Trans. Cybern. 45 (2), 302–315.

Yu, X., Jiang, N., Wang, X., Li, M., 2023. A hybrid algorithm based on grey wolf optimizer and differential evolution for UAV path planning. Expert Syst. Appl. 215, 119327.

Zhang, J., Sanderson, A.C., 2009. JADE: adaptive differential evolution with optional external archive. IEEE Trans. Evol. Comput. 13 (5), 945–958.

Zhu, S., Piotrowski, A.P., Ptak, M., Napiorkowski, J.J., Dai, J., Ji, Q., 2021. How does the calibration method impact the performance of the air2water model for the forecasting of lake surface water temperatures? J. Hydrol. 597, 126219.