Cluster Computing (2025)28:200
https://doi.org/10.1007/s10586-024-04915-4

A novel adjacent matrix-based probabilistic selection mechanism for differential evolution

Rui Zhong[1] · Shilong Zhang[2] · Yujun Zhang[3] · Jun Yu[4]

Received: 17 June 2024 / Revised: 13 November 2024 / Accepted: 14 November 2024

Abstract
This paper proposes a novel selection mechanism for differential evolution (DE) termed the adjacent matrix-based probabilistic (AMP) selection mechanism. When DE enters the selection phase, the proposed method merges the parent and offspring populations and constructs an adjacency matrix using the Manhattan distance. The concept of competition is then introduced, with the probability of being selected as a competitor determined based on the adjacency matrix. According to the proximate optimality principle (POP), closer individuals have a higher probability of being competitors, as neighbors tend to share similar genome and fitness information. This allows a cluster to be represented by a superior individual, reducing redundant fitness evaluations. For each individual, *k* competitors are chosen, and fitness comparisons are conducted. The winner counts are ranked to determine which individuals survive. To evaluate the performance of the proposed AMP selection mechanism, we integrated it with eight variants of DE and conducted numerical experiments on CEC2017 and CEC2022 benchmark functions. The experimental results and statistical analysis confirm the effectiveness and robustness of the AMP selection mechanism, demonstrating its great potential for integration with population-based optimization techniques to address various optimization tasks. The source code of this research can be downloaded from https://github.com/RuiZhong961230/AMP.

Keywords Differential evolution (DE) · Adjacent matrix-based probabilistic (AMP) selection · Proximate optimality principle (POP) · Numerical optimization

1 Introduction

In the rapid development of optimization algorithms, differential evolution (DE) has emerged as a powerful tool for solving complex optimization problems due to its simplicity and effectiveness [1–4]. Inspired by Darwin's theory of evolution, DE relies on the mechanism of mutation, crossover, and selection to evolve solutions over successive generations [5]. In the past decades, numerous variants of DE have been proposed to address various complex optimization problems. However, most of them focus on improving the performance of DE by adjusting the hyperparameters (i.e., scaling factor F, crossover rate Cr, and population size N) [6–14] and introducing efficient search operators [15–21]. At the same time, only a few publications paid attention to the selection mechanism in DE [22–24].

As an important component of DE, the selection mechanism provides the principle to survive individuals to

✉ Jun Yu
yujun@ie.niigata-u.ac.jp

Rui Zhong
zhongrui@iic.hokudai.ac.jp

Shilong Zhang
f24c130g@mail.cc.niigata-u.ac.jp

Yujun Zhang
zhangyj069@gmail.com

1 Information Initiative Center, Hokkaido University, Sapporo, Japan

2 Graduate School of Science and Technology, Niigata University, Niigata, Japan

3 School of New Energy, Jingchu University of Technology, Jingmen, China

4 Institute of Science and Technology, Niigata University, Niigata, Japan

Published online: 21 January 2025

the subsequent generation [25, 26]. Motivated by the "survival of the fittest", it should ensure the survival of elite individuals while maintaining the diversity of the population for exploring the solution space. As one of the simplest and most common selection schemes of DE, the one-to-one greedy selection for the minimization problem is formulated in Eq. (1).

$$\boldsymbol{x}_i^{t+1} = \begin{cases} \boldsymbol{u}_i^t, & \textit{if } f(\boldsymbol{u}_i^t) < f(\boldsymbol{x}_i^t) \\ \boldsymbol{x}_i^t, & \textit{otherwise} \end{cases} \tag{1}$$

where $\boldsymbol{u}_i^t$ and $\boldsymbol{x}_i^t$ denote the constructed offspring individual and parent individual, respectively. This selection mechanism compares each offspring individual with its corresponding parent to survive the individual with better fitness [27, 28]. The benefits of this scheme include (1) simplicity and computational efficiency, (2) steady convergence, and (3) scalability and robustness. However, this conventional approach also faces challenges such as (1) premature optimization, (2) loss of population diversity, and (3) limited ability to escape local optima. Therefore, developing an efficient and effective selection mechanism is necessary and promising to enhance the performance of DE in various optimization domains.

This paper proposes a novel adjacency matrix-based probabilistic (AMP) selection mechanism and integrates it with variants of DE to further investigate its adaptability and scalability. Specifically, after generating the offspring population, AMP merges it with the parent population. The Manhattan distance between each pair of individuals is calculated and stored in the adjacency matrix. For the current individual $\boldsymbol{x}_i^t$, AMP then randomly selects k competitor individuals based on the Manhattan distance, with closer individuals having a higher probability of being selected. The fitness of the current individual $\boldsymbol{x}_i^t$ is compared with the competitors in sequence, and the counts where $\boldsymbol{x}_i^t$ outperforms a competitor are recorded. Finally, AMP survives the individuals based on the number of counts. To evaluate the performance of our proposed AMP selection mechanism, we integrate it with conventional DE with DE/rand/1, DE/cur/1, and DE/cur-to-best/1 mutation strategies, JADE, and competitive DE (CDE). The numerical experiments are conducted on IEEE-CEC2017 and CEC2022 benchmark functions to confirm the effectiveness and adaptability of our proposed AMP selection mechanism.

The remainder of this paper is organized as follows: Sect. 2 presents the related works including the basic framework of DE and the progress on selection mechanisms in DE. Section 3 presents the details of our proposed AMP selection mechanism. The detailed numerical experiments and performance analysis are presented in Sect. 4. Finally, Sect. 6 concludes this paper.

2 Related works

2.1 Basic framework of DE

DE is a population-based stochastic optimization algorithm proposed by Storn and Price in 1995 [5], which belongs to the family of evolutionary algorithms (EAs) and mainly consists of population initialization, mutation, crossover, and selection.

Population initialization: Similar to most population-based EAs, the first phase of DE is population initialization, as formulated in Eq. (2).

$$X = \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \boldsymbol{x}_3 \\ \vdots \\ \boldsymbol{x}_N \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1D} \\ x_{21} & x_{22} & \cdots & x_{2D} \\ x_{31} & x_{32} & \cdots & x_{3D} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{ND} \end{bmatrix} \tag{2}$$
$$x_{ij} = r \cdot (\boldsymbol{ub}_j - \boldsymbol{lb}_j) + \boldsymbol{lb}_j$$

where X represents the population, $\boldsymbol{x}_i$ is the i^{th} individual, and x_{ij} represents the value in the j^{th} dimension of the $\boldsymbol{x}_i$. $\boldsymbol{lb}_j$ and $\boldsymbol{ub}_j$ are the lower and the upper bound of the j^{th} dimension, respectively, and r is a random number following a uniform distribution in $U(0, 1)$.

Mutation schemes: Subsequently, DE utilizes the mutation scheme to construct mutant vectors. Equation (3) summarizes some popular and representative mutation schemes in DE.

$$\begin{aligned} &\text{DE/rand/1}: \boldsymbol{v}_i^t = \boldsymbol{x}_{r1}^t + F \cdot (\boldsymbol{x}_{r2}^t - \boldsymbol{x}_{r3}^t) \\ &\text{DE/cur/1}: \boldsymbol{v}_i^t = \boldsymbol{x}_i^t + F \cdot (\boldsymbol{x}_{r1}^t - \boldsymbol{x}_{r2}^t) \\ &\text{DE/best/1}: \boldsymbol{v}_i^t = \boldsymbol{x}_{best}^t + F \cdot (\boldsymbol{x}_{r1}^t - \boldsymbol{x}_{r2}^t) \\ &\text{DE/cur-to-best/1}: \boldsymbol{v}_i^t = \boldsymbol{x}_i^t + F \cdot (\boldsymbol{x}_{best}^t - \boldsymbol{x}_i^t) + F \cdot (\boldsymbol{x}_{r1}^t - \boldsymbol{x}_{r2}^t) \\ &\text{DE/cur-to- pbest/1}: \boldsymbol{v}_i^t = \boldsymbol{x}_i^t + F \cdot (\boldsymbol{x}_{pbest}^t - \boldsymbol{x}_i^t) + F \cdot (\boldsymbol{x}_{r1}^t - \boldsymbol{x}_{r2}^t) \end{aligned} \tag{3}$$

where $\boldsymbol{v}_i^t$ is the constructed mutant vector, $r1$, $r2$, and $r3$ are three mutually different integers selected from $\{1, 2, 3, \ldots, N\}$ randomly, and F is the scaling factor in the range of (0, 2] [29].

Crossover: The crossover operator in DE combines the mutant vector with the target vector to generate a trial vector, which is responsible for the exploitative search in the DE framework. The mathematical model of the most common binomial crossover is presented in Eq. (4).

$$\boldsymbol{v}_{i,j}^t = \begin{cases} \boldsymbol{u}_{i,j}^t, & \textit{if } r \le Cr \textit{ or } j = j_{rand} \\ \boldsymbol{x}_{i,j}^t, & \textit{otherwise} \end{cases} \tag{4}$$

where Cr in (0, 1) is the crossover rate to control the probability of inheritance between the mutant vector $\boldsymbol{u}_{i,j}^t$ and the target vector $\boldsymbol{x}_{i,j}^t$. j_{rand} is a random integer in $\{1, 2, 3, ..., D\}$.

Selection: The one-to-one greedy selection mechanism in DE is formulated in Eq. (5).

$$\boldsymbol{x}_i^{t+1} = \begin{cases} \boldsymbol{u}_i^t, & if\ f(\boldsymbol{u}_i^t) \leq f(\boldsymbol{x}_i^t) \\ \boldsymbol{x}_i^t, & otherwise \end{cases} \quad (5)$$

In summary, the pseudocode of the basic DE is presented in Algorithm 1.

Algorithm 1 Basic DE

Require: Population size:N, Dimension:D, Maximum iteration:T
Ensure: Optimum: $\boldsymbol{x}_{best}^t$

```
Population and parameters initialization
t = 0
x^t_best ← best(X)
while t < T do
    for i = 0 to N do
        Construct the mutant vector using Eq. (3)
        Crossover using Eq. (4)
    end for
    Selection using Eq. (5)
    x^t_best ← best(X)
    t ← t + 1
end while
return x^t_best
```

the trial vector without any comparison. **Closest Survival Selection (CC)**: The trial vector is always compared with the closest individual to it in the population. **Hybrid Survival Selection (CH)**: This method combines the closest survival selection and the worst survival selection. These selection mechanisms were integrated into DE, and numerical experiments with statistical analysis confirmed that CW can accelerate optimization, while CC enhances the robustness of DE.

2.2 Progress on selection mechanisms in DE

The selection mechanism in DE plays a crucial role in determining which individuals survive to form the next generation. This section presents an overview of the progress in selection mechanisms in DE.

Padhye et al. [30] and Vinh et al. [31] introduced the elitist selection mechanism to DE. After constructing the offspring population, it is merged with the current population, and only the top-k individuals survive, where k equals the population size. This high-level greedy selection mechanism accelerates optimization for problems with a simple fitness landscape but may trap the algorithm in local optima and lead to rapid stagnation in complex optimization problems.

Tagawa et al. [32] developed five criteria for the selection mechanism: **Family Survival Selection (CF)**: The worse target vector is immediately replaced by the corresponding trial vector. **Worst Survival Selection (CW)**: The trial vector is always compared with the worst individual in the population. **Absolute Survival Selection (CA)**: The worst individual in the population is replaced by the trial vector without any comparison.

Arka et al. [33] focused on the optimization problems in noisy environments and proposed a distance-based selection mechanism formulated in Eq. (6).

$$\boldsymbol{x}_i^{t+1} = \begin{cases} \boldsymbol{u}_i^t, & if\ f(\boldsymbol{u}_i^t) \leq f(\boldsymbol{x}_i^t) \\ \boldsymbol{u}_i^t, & if\ f(\boldsymbol{u}_i^t) > f(\boldsymbol{x}_i^t)\ and\ p_s \leq e^{-\frac{\Delta f}{Dist}} \\ \boldsymbol{x}_i^t, & otherwise \end{cases} \quad (6)$$

where p_s is a random number following a uniform distribution $U(0, 1)$, $\Delta f = |f(\boldsymbol{u}_i^t) - f(\boldsymbol{x}_i^{t+1})|$, and $Dist = \sum_{j=0}^{k} |\boldsymbol{u}_{i,j}^t - \boldsymbol{x}_{i,j}^t|$. Considering that the fitness landscape is corrupted by noise, the one-to-one greedy selection method suffers from uncertainty, making it nearly impossible to determine whether the offspring individual is definitively better or worse than the parent. To address this, the proposed distance-based selection mechanism incorporates a probability that even an inferior offspring can survive occasionally.

Tian et al. [34] proposed a novel diversity-based selection strategy as formulated in Eq. (7).

$$x_i^{t+1} = \begin{cases} u_i^t, & \text{if } f(u_i^t) \le f(x_i^t) \\ u_i^t, & \text{if } f_w(u_i^t) \le f_w(x_i^t) \text{ and } x_i^t \ne x_{best}^t \\ x_i^t, & \text{otherwise} \end{cases} \tag{7}$$

where $f_w(\cdot)$ is the corrected fitness function and defined in Eq. (8).

$$f_w(x_i^t) = \alpha \frac{f(x_i^t) - f_{min}}{f_{max} - f_{min}} + (1 - \alpha) \frac{Dist_{max} - Dist(x_{best}^t, x_i^t)}{Dist_{max} + Dist(x_{best}^t, x_i^t)} \tag{8}$$

α is a weight factor to balance the proportion between the fitness and distance as suggested in [0.8, 1] [34]. f_{max} and f_{min} are the maximum and the minimum fitness values in the current and the offspring population, respectively. $Dist_{max}$ is the maximum Euclidean distance between the best individual x_{best}^t and other individuals in both the current and the offspring population, and $Dist(x_{best}^t, x_i^t)$ calculates the Euclidean distance between the best individual x_{best}^t and the current individual x_i^t.

To a pair of the offspring individual u_i^t and the current individual x_i^t, the diversity-based selection mechanism allows u_i^t to survive with a certain probability, even if it has a worse fitness value but is far from the current best individual x_{best}^t. This strategy simultaneously considers the fitness value and the distance factor and can maintain the population diversity during optimization.

Zeng et al. [35] proposed a hybrid selection mechanism in DE. Specifically, when an individual is not in a state of stagnation, the proposed hybrid selection functions as same as the classical selection operator that chooses the best vector between the trial vector and the parent vector to survive. However, when the individual is in a state of stagnation, three other candidate vectors may be selected for the next generation:

- The best individual from all the discarded trial vectors of the parent vector.
- The second-best individual from all the discarded trial vectors of the parent vector.
- A randomly chosen individual from all the successfully updated solutions.

Several DE variants were equipped with the proposed hybrid selection mechanism, and the numerical experiments confirmed that it can significantly accelerate optimization.

Although previous works have considered distance information between parent and offspring individuals, as well as probabilistic survival, these selection mechanisms rely on limited information sources, which restrict decision-making diversity and further reduce the population diversity. In this paper, we propose an AMP selection mechanism that considers the distance information of the entire population and introduces a competitive mechanism into the selection process. Elite individuals are preserved through competition with other individuals, while inferior individuals still have a chance of being accepted. This strategy effectively captures both natural selection and random chance, reflecting the balance of survival of the fittest with occasional opportunities for weaker individuals in nature.

3 Our proposal: AMP selection mechanism

This section introduces our proposed AMP selection mechanism in detail. Section 3.1 presents the overview of the AMP selection mechanism, Sect. 3.2 introduces the AMP selection mechanism in detail, and the computational complexity of the AMP selection mechanism is analyzed in Sect. 3.3.

3.1 Overview of the AMP selection mechanism

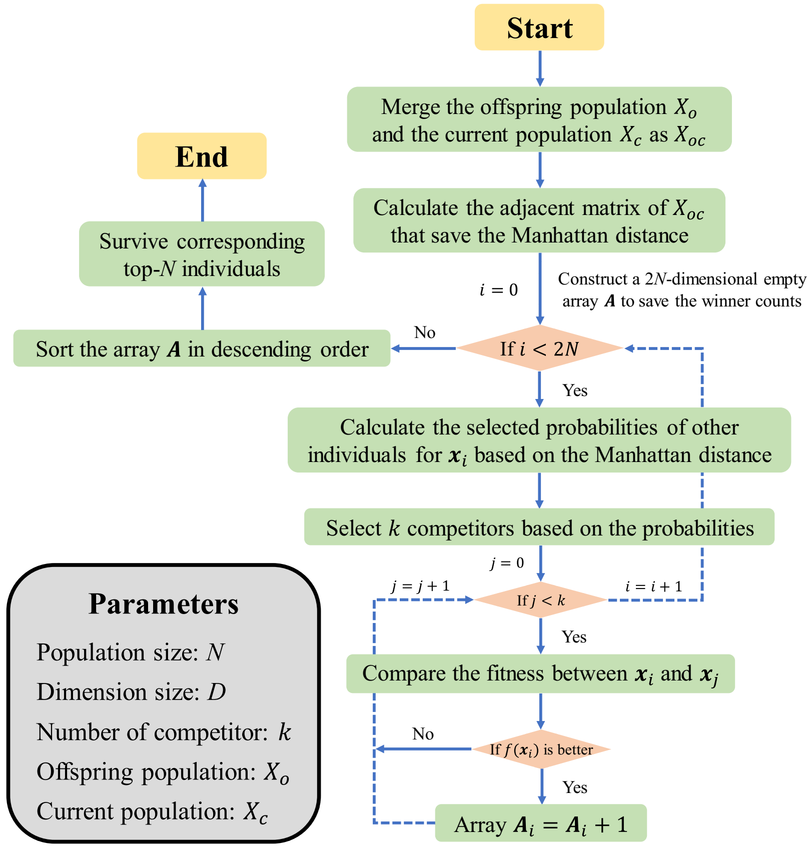
The flowchart of the AMP selection mechanism is presented in Fig. 1. We first define the input parameters including the population size N, the dimension size D, the number of competitors k, the offspring population X_o, and the current population X_c. At the beginning of the selection phase, the offspring population and the current population are merged to form X_{oc}. Next, the Manhattan distance between every pair of individuals is calculated and stored in the adjacent matrix. A $2N$-dimensional array A is initialized to save the winner counts for the corresponding individuals.

When the AMP selection enters the main loop, it calculates the selected probabilities of other individuals concerning the current individual x_i based on the Manhattan distance saved in the adjacent matrix. Subsequently, k competitor individuals are chosen based on the probabilities and compared with the current individual x_i, and the winner counts are saved in the corresponding dimension of the array A. After all individuals are compared, AMP exits the loop, sorts the array A in descending order, and the top-N individuals survive to the next generation. In the following context, we will introduce the details of the proposed AMP selection mechanism.

3.2 Details of the AMP selection mechanism

This section introduces the details of the AMP selection mechanism. Section 3.2.1 presents the adjacent matrix construction, Sect. 3.2.2 describes the selective probability definition, and the survival individual determination is introduced in Sect. 3.2.3.

Fig. 1 The flowchart of the AMP selection mechanism



3.2.1 Adjacent matrix construction

The initial step of the adjacent matrix construction is to merge the offspring population X_o and the current population X_c. A demonstration is presented in Fig. 2.

We simply merge these two population and their fitness in order. Then, the Manhattan distance between every pair of individuals is calculated and saved in the adjacent matrix, which is demonstrated in Eq. (9).

$$\mathbf{AM} = \begin{bmatrix} NaN & Dist_{1,2} & \cdots & Dist_{1,2N} \\ Dist_{2,1} & NaN & \cdots & Dist_{2,2N} \\ Dist_{3,1} & Dist_{3,2} & \cdots & Dist_{3,2N} \\ \vdots & \vdots & \ddots & \vdots \\ Dist_{2N,1} & Dist_{2N,2} & \cdots & NaN \end{bmatrix} \tag{9}$$

$$Dist_{i,j} = \sum_{k=1}^{D} |\boldsymbol{x}_{i,k}^t - \boldsymbol{x}_{j,k}^t|$$

Here, $Dist_{i,j}$ in the adjacent matrix **AM** denotes the Manhattan distance between individuals $\boldsymbol{x}_i^t$ and $\boldsymbol{x}_j^t$. Since the element in diagonal means the Manhattan distance between the specific individual and itself, which is meaningless, we manually fill them with not a number NaN. Additionally, $Dist_{i,j} = Dist_{j,i}$ since the representation is the same.

3.2.2 Selective probability definition

We first present the relationship between the probability function and Manhattan distance in Eq. (10).

$$C(\boldsymbol{x}_i^t, \boldsymbol{x}_j^t) = \frac{1}{Dist_{i,j}},\ j \in \{1, 2, ...N, N+1, ..., 2N\}\ and\ j \neq i$$
$$P(\boldsymbol{x}_i^t, \boldsymbol{x}_j^t) = \frac{C(\boldsymbol{x}_i^t, \boldsymbol{x}_j^t)}{\sum_{k=1}^{2N} C(\boldsymbol{x}_i^t, \boldsymbol{x}_k^t)} \tag{10}$$

where $\boldsymbol{x}_i^t$ and $\boldsymbol{x}_j^t$ are the current and competitor individuals, respectively. $C(\boldsymbol{x}_i^t, \boldsymbol{x}_j^t)$ denotes the corrected function to measure the fitness based on the distance between $\boldsymbol{x}_i^t$ and $\boldsymbol{x}_j^t$, and $P(\boldsymbol{x}_i^t, \boldsymbol{x}_j^t)$ defines the probability of the current $\boldsymbol{x}_i^t$ to select $\boldsymbol{x}_j^t$ as the competitor.

The corrected fitness function $C(\boldsymbol{x}_i^t, \boldsymbol{x}_j^t)$ is defined based on a hypothesis that the closer individuals have a higher similarity in genome and fitness information as suggested in the proximate optimality principle (POP) [36, 37]. This redundant information may waste the

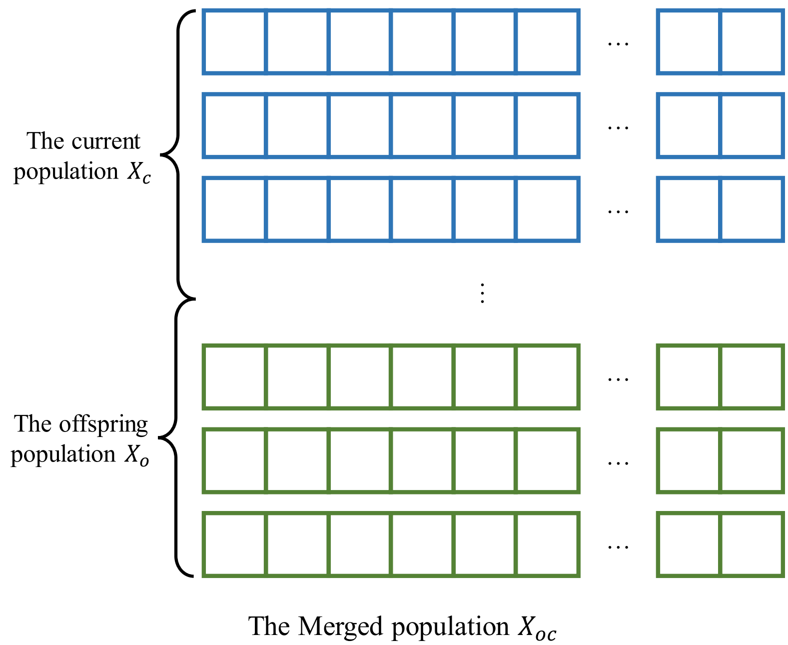


Fig. 2 A demonstration of the population merge

fitness evaluation and further decelerate optimization convergence. Therefore, the corrected fitness function $C(\boldsymbol{x}_i^t, \boldsymbol{x}_j^t)$ is defined as the inverse function of the Manhattan distance, demonstrating that the closer individuals of $\boldsymbol{x}_i^t$ have a higher probability of being selected as competitors. In this situation, the neighbors of $\boldsymbol{x}_i^t$ are preferred to be selected as competitors, and the superior individuals will survive and represent this region. Relatively, the individual who has a worse fitness value but is far away from the $\boldsymbol{x}_i^t$ also has a survival probability.

3.2.3 Survival individual determination

After the selective probabilities of other individuals for the current individual $\boldsymbol{x}_i^t$ are determined, the AMP selection mechanism selects k competitors based on the probabilities. Subsequently, the fitness comparison between $\boldsymbol{x}_i^t$ and competitors is conducted using Algorithm 2.

The winner counts between $\boldsymbol{x}_i^t$ and competitors are saved in the corresponding dimension of the array $\boldsymbol{A}_i$. After all the individuals are compared and the main loop is finished, the AMP selection mechanism sorts the array $\boldsymbol{A}$ and survives the top-N individuals, as demonstrated in Fig. 3.

In summary, the pseudocode of the proposed AMP selection mechanism is presented in Algorithm 3.

3.3 Computational complexity of the AMP selection mechanism

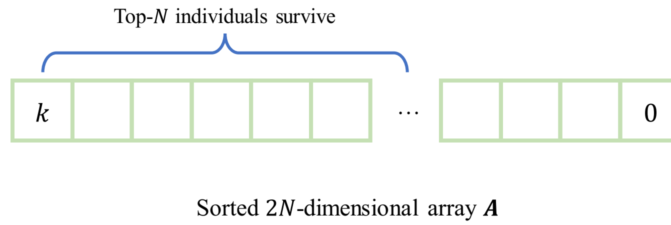
This section analyzes the computational complexity of the AMP selection mechanism following Algorithm 3 theoretically. In Algorithm 3 Line 2, the adjacent matrix based on the Manhattan distance is constructed, and the computational complexity is $O(2N \cdot 2N \cdot D) := O(N^2 \cdot D)$. In Algorithm 3 from Line 5 to 7, the computational complexity of the selective probability calculation for $\boldsymbol{x}_i^t$ in Line 5 is $O(2N)$, and the fitness comparison between $\boldsymbol{x}_i^t$ and competitors in Line 7 is $O(k)$. Given that these operators are in the loop, and the complete computational complexity is $O(2N \cdot (2N + k))$. Since k must be smaller than $2N$, and

Algorithm 2 Fitness comparison

Require: Current individual: $\boldsymbol{x}_i^t$, Array: $\boldsymbol{A}$, k-Competitors: C^t

for $j = 0$ *to* k **do**
    if $\boldsymbol{x}_i^t$ has a better fitness than C_j^t **then**
        $\boldsymbol{A}_i = \boldsymbol{A}_i + 1$
    end if
end for

Fig. 3 A demonstration of the top-N selection based on the sorted array $\boldsymbol{A}$



Algorithm 3 AMP selection

Require: Population size: N, Dimension: D, Number of competitor: k, Offspring population: X_o, Current population: X_c
Ensure: Survival individuals: X_s
1: Merge X_o and X_c to form X_{oc}
2: Calculate the adjacent matrix using Eq. (9)
3: Construct a $2N$-dimensional empty array $\boldsymbol{A}$
4: **for** $i = 0$ *to* $2N$ **do**
5: Calculate the selective probabilities for $\boldsymbol{x}_i^t$ using Eq. (10)
6: Select k competitors for $\boldsymbol{x}_i^t$ based on the probabilities
7: Compare $\boldsymbol{x}_i^t$ with competitors and update $\boldsymbol{A}_i$ using Algorithm 2
8: **end for**
9: Sort the array $\boldsymbol{A}$ in descending order
10: Survive top-N individuals as X_s
11: **return** X_s

$O(2N \cdot (2N + k))$ can be further simplified to $O(N^2)$. Additionally, the computational complexity of the sort operator in Line 9 is $O(N \log N)$. Therefore, the computational complexity of the AMP selection mechanism is presented in Eq. (11).

$$O(N^2 \cdot D + N^2 + N \log N) := O(N^2 \cdot D) \tag{11}$$

where the computational complexity of the conventional selection mechanism is $O(N)$. Considering the significant performance improvement of DE by integrating the AMP selection mechanism and this research mainly focuses on small- and middle-scale optimization problems, the additional computational consumption is affordable.

4 Numerical experiments

This section presents numerical experiments to evaluate our proposed AMP selection mechanism. We integrate DE and its well-known variants with our proposed AMP selection mechanism and conduct optimization experiments on low- and median-scale IEEE-CEC2017 and CEC2022 benchmark functions. Section 4.1 summarizes the experimental settings including experimental environments, benchmark functions, and algorithm parameter settings. Section 4.2 presents the experimental results and performance analysis.

4.1 Experimental settings

4.1.1 Experimental environments and implementation

We conduct numerical experiments using Python 3.11, executed on a Lenovo Legion R9000P running Windows 11. The configuration of system hardware includes an AMD Ryzen 7 5800 H processor with Radeon Graphics, clocked at 3.20 GHz, and 16GB of RAM.

4.1.2 Benchmark functions

We adopt 10- and 30-dimensional IEEE-CEC2017[1] and 10- and 20-dimensional CEC2022[2] as benchmarks to evaluate the performance of our proposed AMP selection mechanism. These benchmarks are provided by the Opfunu library [38]. Tables 1 and 2 summarize the basic information of these two benchmarks. Since f_2 in CEC2017 is disabled, f_{30} serves as a compensation.

---

[1] https://github.com/P-N-Suganthan/CEC2017-BoundContrained

[2] https://github.com/P-N-Suganthan/2022-SO-BO

Table 1 Summary of the CEC2017 benchmark functions

No.	Func.	Feature	Optimum
f_1	Shifted and Rotated Bent Cigar function	Uni.	100
f_3	Shifted and Rotated Rosenbrock's function	Multi.	300
f_4	Shifted and Rotated Rastrigin's function		400
f_5	Shifted and Rotated Expanded Scaffer's F6 function		500
f_6	Shifted and Rotated Lunacek Bi_Rastrigin function		600
f_7	Shifted and Rotated Non-Continuous Rastrigin's function		700
f_8	Shifted and Rotated Levy function		800
f_9	Shifted and Rotated Schwefel's function		900
f_{10}	Hybrid function 1 (N = 3)	Hybrid.	1000
f_{11}	Hybrid function 2 (N = 3)		1100
f_{12}	Hybrid function 3 (N = 3)		1200
f_{13}	Hybrid function 4 (N = 4)		1300
f_{14}	Hybrid function 5 (N = 4)		1400
f_{15}	Hybrid function 6 (N = 4)		1500
f_{16}	Hybrid function 6 (N = 5)		1600
f_{17}	Hybrid function 6 (N = 5)		1700
f_{18}	Hybrid function 6 (N = 5)		1800
f_{19}	Hybrid function (N = 6)		1900
f_{20}	Composition function 1 (N = 3)	Comp.	2000
f_{21}	Composition function 2 (N = 3)		2100
f_{22}	Composition function 3 (N = 4)		2200
f_{23}	Composition function 4 (N = 4)		2300
f_{24}	Composition function 5 (N = 5)		2400
f_{25}	Composition function 6 (N = 5)		2500
f_{26}	Composition function 7 (N = 6)		2600
f_{27}	Composition function 8 (N = 6)		2700
f_{28}	Composition function 9 (N = 3)		2800
f_{29}	Composition function 10 (N = 3)		2900
f_{30}	Composition function 11 (N = 3)		3000
Search range: $[-100, 100]^D$			

Uni.=Unimodal function, Multi.=Simple multimodal function, Hybrid.=Hybrid function, Comp.=Composition function

4.1.3 Parameters of algorithms

In this study, we integrate the proposed AMP selection mechanism into the variants of DE, as summarized in Table 3.

The parameter settings of optimizers follow the optimal configuration in the corresponding publications. We integrate these DE-based algorithms with our proposed AMP selection mechanism to form DE_{rand}-AMP, DE_{cur}-AMP, DE_{best}-AMP, $DE_{cur2best}$-AMP, $DE_{cur2pbest}$-AMP, JADE-AMP, GTDE-AMP, and CDE-AMP. The only difference is that we replace the one-to-one greedy selection mechanism in the basic version of these algorithms with the AMP selection mechanism. Besides, the population size of all algorithms is fixed at 100, maximum fitness evaluation (FE_{max}) is set to 1000 $\times$ dimension size, and the number of competitors is set to 10. Each algorithm is independently repeated 30 times to alleviate the effect of randomness.

4.2 Experimental results and performance analysis

This section summarizes the experimental results and statistical analysis of DE-based optimizers on IEEE-CEC2017 and CEC2022 benchmark functions. To identify the significance of differences between DE_{rand} and DE_{rand}-AMP, DE_{cur} and DE_{cur}-AMP, and other combinations, we employ the Wilcoxon test, and symbols $+$, $\approx$, and $-$ represent that our proposal is significantly better, without significance, and significantly worse than the basic DE variants, and the best fitness in each comparison is in bold.

Table 2 Summary of the CEC2022 benchmark functions

Func.	Description	Feature	Optimum
f_1	Shifted and full Rotated Zakharov	Uni.	300
f_2	Shifted and full Rotated Rosenbrock	Basic.	400
f_3	Shifted and full Rotated Expanded Schaffer f_6		600
f_4	Shifted and full Rotated Non-Continuous Rastrigin		800
f_5	Shifted and full Rotated Levy		900
f_6	Hybrid function 1 (N = 3)	Hybrid.	1800
f_7	Hybrid function 2 (N = 6)		2000
f_8	Hybrid function 3 (N = 5)		2200
f_9	Composition function 1 (N = 5)	Comp.	2300
f_{10}	Composition function 2 (N = 4)		2400
f_{11}	Composition function 3 (N = 5)		2600
f_{12}	Composition function 3 (N = 6)		2700
Search range: $[-100, 100]^D$			

Uni.=Unimodal function, Basic.=Basic function, Hybrid.=Hybrid function, Comp.=Composition function

Table 3 The parameters of the variants of DE

Alg.	Parameters	Value
DE_{rand} [5]	F and Cr	0.8 and 0.7
	Mutation scheme	DE/rand/1
DE_{cur} [5]	F and Cr	0.8 and 0.7
	Mutation scheme	DE/cur/1
DE_{best} [5]	F and Cr	0.8 and 0.7
	Mutation scheme	DE/best/1
$DE_{cur2best}$ [5]	F and Cr	0.8 and 0.7
	Mutation scheme	DE/cur-to-best/1
$DE_{cur2pbest}$ [39]	F and Cr	0.8 and 0.7
	Mutation scheme	DE/cur-to-*p*best/1
JADE [39]	μ_F and μ_{Cr}	0.5 and 0.5
	σ_F and σ_{Cr}	0.1 and 0.1
	Mutation scheme	DE/cur-to-*p*best/1
GTDE [40]	μ_F and σ_F	0.7 and 0.5
	μ_{Cr} and σ_{Cr}	0.5 and 0.5
CDE [41]	Scaling factor F	$N(0.5, 0.3)$
	Crossover rate Cr	$N(0.5, 0.3)$
	Mutation scheme	DE/winner-to-best/1

4.2.1 Experimental results on CEC2017

Tables 4, 5, 6, and 7 summarize the experimental results and statistical analysis on CEC2017. Figures 4 and 5 present the convergence curves of optimizers on representative functions of CEC2017 (i.e., f_1: Unimodal function; f_3, f_4 and f_7: Simple multimodal functions; f_{10}, f_{12}, f_{15}, and f_{18}: Hybrid functions; f_{22}, f_{25}, f_{26}, and f_{30}: Composite functions.).

The summarized experimental results and statistical analysis confirm the effectiveness and robustness of our proposed AMP selection mechanism in various DE-based optimizers. Significant deterioration occurs only several times. Additionally, the statistical significance summary between DE_{best} and DE_{best}-AMP is 15/11/3 in 10-D, where DE_{best}-AMP is significantly better than DE_{best} in fifteen instances, shows no significant difference in eleven instances and is significantly worse than DE_{best} in three instances. Compared to other DE-based optimization algorithms, the proposed AMP selection mechanism is relatively less well-adapted with DE_{best}.

Additionally, as the dimension of the problem increases to the 30 dimensions, the degeneration situations are observed when the proposed AMP selection mechanism is integrated with DE_{best}, $DE_{cur2best}$, $DE_{cur2pbest}$, JADE, and CDE. This phenomenon indicates that our proposed AMP selection mechanism may not perform well in high-dimensional search space. We infer that due to the existence of the curse of dimensionality, the search domain increases exponentially as the dimension size increases linearly, which affects the efficiency of the AMP selection mechanism and further decelerates the optimization convergence.

In summary, the significance, robustness, and overall superiority of the AMP selection mechanism cannot be neglected, and we believe that this competitive selection mechanism can be extended and integrated with various optimization algorithms.

Table 4 The experimental results and statistical analysis on 10-dimensional CEC2017 benchmark functions

Func.		DE_{rand}	DE_{rand}-AMP	DE_{cur}	DE_{cur}-AMP	DE_{best}	DE_{best}-AMP	$DE_{cur2best}$	$DE_{cur2best}$-AMP	$DE_{cur2pbest}$	$DE_{cur2pbest}$-AMP	JADE	JADE-AMP	GTDE	GTDE-AMP	CDE	CDE-AMP
f_1	mean	2.179e+08 +	**8.605e+07**	6.481e+08 +	**6.050e+07**	2.062e+06 +	**4.454e+05**	2.360e+06 +	**4.583e+05**	2.043e+06 +	**2.765e+05**	1.047e+05 +	**2.293e+03**	1.209e+04 +	**3.558e+03**	3.464e+03 +	**1.805e+03**
	std	5.787e+07	3.110e+07	1.206e+08	1.746e+07	2.388e+06	5.551e+05	2.646e+06	2.736e+05	8.706e+05	1.319e+05	6.082e+04	1.711e+03	1.195e+04	3.140e+03	2.680e+03	2.331e+03
f_3	mean	2.311e+04 +	**1.993e+04**	2.073e+04 +	**1.882e+04**	7.509e+03 +	**4.603e+03**	7.575e+03 +	**4.573e+03**	5.552e+03 +	**3.823e+03**	1.255e+03 +	**3.431e+02**	2.184e+03 +	**3.139e+02**	2.773e+03 +	**3.478e+02**
	std	4.069e+03	6.613e+03	2.524e+03	4.332e+03	2.557e+03	1.155e+03	1.837e+03	7.118e+02	1.197e+03	1.004e+03	1.073e+03	4.588e+01	1.449e+03	1.264e+01	1.453e+03	3.086e+01
f_4	mean	4.312e+02 +	**4.093e+02**	4.771e+02 +	**4.087e+02**	**4.062e+02** −	4.094e+02	4.123e+02 +	**4.039e+02**	4.052e+02 +	**4.037e+02**	4.060e+02 +	**4.056e+02**	4.115e+02 +	**4.041e+02**	4.054e+02 +	**4.047e+02**
	std	7.864e+00	4.999e−01	1.943e+01	7.230e−01	8.767e−01	1.543e+01	1.990e+01	5.430e−−01	2.672e−01	4.050e−01	6.209e−01	7.939e−01	1.509e+01	6.515e−01	1.315e+00	1.937e+00
f_5	mean	5.461e+02 ≈	**5.430e+02**	5.562e+02 +	**5.441e+02**	5.429e+02 ≈	**5.420e+02**	5.390e+02 ≈	**5.377e+02**	5.388e+02 ≈	**5.364e+02**	5.257e+02 +	**5.158e+02**	5.328e+02 +	**5.210e+02**	5.317e+02 +	**5.096e+02**
	std	9.701e+00	5.197e+00	4.919e+00	8.845e+00	5.503e+00	5.117e+00	5.838e+00	3.973e+00	5.022e+00	5.794e+00	1.731e+00	6.914e+00	5.378e+00	9.801e+00	3.627e+00	5.118e+00
f_6	mean	6.150e+02 +	**6.069e+02**	6.341e+02 +	**6.087e+02**	6.021e+02 +	**6.004e+02**	6.017e+02 +	**6.003e+02**	6.009e+02 +	**6.001e+02**	6.003e+02 +	**6.000e+02**	6.001e+02 +	**6.000e+02**	6.000e+02 +	**6.000e+02**
	std	3.310e+00	1.105e+00	4.833e+00	1.688e+00	5.362e−01	6.509e−02	3.581e−01	1.206e−01	3.003e−01	1.864e−02	9.873e−02	5.216e−04	5.807e−02	2.249e−03	8.771e−03	4.636e−05
f_7	mean	7.791e+02 +	**7.730e+02**	8.127e+02 +	**7.707e+02**	7.580e+02 ≈	**7.554e+02**	7.582e+02 +	**7.504e+02**	7.496e+02 ≈	**7.471e+02**	7.368e+02 +	**7.297e+02**	7.452e+02 +	**7.263e+02**	7.422e+02 +	**7.266e+02**
	std	6.537e+00	4.470e+00	9.103e+00	8.052e+00	7.401e+00	4.076e+00	7.805e+00	3.883e+00	5.628e+00	3.845e+00	4.229e+00	2.634e+00	1.310e+01	9.839e+00	4.903e+00	7.533e+00
f_8	mean	8.513e+02 +	**8.531e+02**	8.605e+02 +	**8.527e+02**	8.466e+02 +	**8.425e+02**	8.442e+02 ≈	**8.431e+02**	8.409e+02 +	**8.318e+02**	8.249e+02 +	**8.130e+02**	8.327e+02 +	**8.154e+02**	8.270e+02 +	**8.064e+02**
	std	9.044e+00	4.236e+00	1.082e+01	5.709e+00	3.803e+00	7.109e+00	3.966e+00	6.386e+00	4.802e+00	4.635e+00	4.612e+00	3.800e+00	7.777e+00	7.225e+00	5.469e+00	2.397e+00
f_9	mean	1.268e+03 +	**1.018e+03**	1.769e+03 +	**1.016e+03**	9.167e+02 +	**9.004e+02**	9.093e+02 +	**9.003e+02**	9.021e+02 +	**9.000e+02**	9.035e+02 +	**9.000e+02**	9.003e+02 ≈	**9.002e+02**	9.000e+02 +	**9.000e+02**
	std	9.918e+01	3.139e+01	3.014e+02	4.068e+01	1.323e+01	3.302e−01	5.318e+00	9.787e−02	7.939e−01	2.419e−02	9.597e+00	3.653e−07	2.924e−01	2.227e−01	1.281e−02	3.314e−03
f_{10}	mean	2.290e+03 +	**2.046e+03**	2.449e+03 +	**1.882e+03**	1.989e+03 +	**1.927e+03**	2.382e+03 +	**2.117e+03**	2.545e+03 +	**2.490e+03**	2.242e+03 +	**1.777e+03**	2.046e+03 +	**1.522e+03**	2.403e+03 +	**1.797e+03**
	std	1.068e+02	1.475e+02	1.307e+02	2.951e+02	3.139e+02	3.841e+02	2.640e+02	2.638e+02	7.910e+01	1.410e+02	1.489e+02	2.869e+02	2.990e+02	2.993e+02	1.749e+02	4.475e+02
f_{11}	mean	1.190e+03 +	**1.128e+03**	1.241e+03 +	**1.132e+03**	1.122e+03 +	**1.113e+03**	1.143e+03 +	**1.114e+03**	1.115e+03 +	**1.110e+03**	1.110e+03 +	**1.105e+03**	1.113e+03 ≈	**1.113e+03**	1.107e+03 ≈	**1.106e+03**
	std	1.435e+01	8.544e+00	2.895e+01	5.687e+00	4.971e+00	2.619e+00	6.712e+01	1.706e+00	2.599e+00	1.904e+00	1.512e+00	2.538e+00	3.104e+00	1.040e+01	1.189e+00	3.401e+00
f_{12}	mean	1.194e+07 ≈	**1.032e+07**	1.558e+07 +	**6.098e+06**	8.779e+05 +	**2.007e+05**	9.670e+05 +	**7.263e+05**	9.793e+05 +	**6.115e+05**	9.893e+04 +	**9.254e+03**	4.276e+04 +	**1.525e+04**	2.207e+05 +	**9.752e+03**
	std	6.059e+06	3.660e+06	6.447e+06	2.167e+06	9.422e+05	2.673e+05	9.634e+05	3.948e+05	5.391e+05	2.725e+05	9.489e+04	4.614e+03	3.270e+04	6.649e+03	2.124e+05	6.594e+03
f_{13}	mean	2.545e+03 +	**1.633e+03**	5.072e+03 +	**1.603e+03**	1.406e+03 ≈	**1.359e+03**	1.382e+03 ≈	**1.357e+03**	1.361e+03 +	**1.339e+03**	1.436e+03 +	**1.323e+03**	1.413e+03 +	**1.345e+03**	1.469e+03 +	**1.343e+03**
	std	3.008e+02	1.196e+02	1.803e+03	9.592e+01	1.005e+02	4.060e+01	2.740e+01	1.833e+01	1.036e+01	7.850e+00	5.898e+01	5.962e+00	2.760e+01	4.150e+01	9.077e+01	3.087e+01
f_{14}	mean	1.440e+03 +	**1.433e+03**	1.477e+03 +	**1.436e+03**	1.430e+03 ≈	**1.430e+03**	1.430e+03 ≈	**1.430e+03**	1.431e+03 +	**1.427e+03**	1.430e+03 +	**1.422e+03**	1.439e+03 +	**1.428e+03**	1.427e+03 +	**1.411e+03**
	std	4.187e+00	3.685e+00	1.231e+01	1.961e+00	3.802e+00	3.034e+00	2.855e+00	1.581e+00	2.539e+00	1.782e+00	4.520e+00	4.488e+00	1.232e+01	5.875e+00	3.368e+00	6.159e+00
f_{15}	mean	1.570e+03 +	**1.529e+03**	1.769e+03 +	**1.535e+03**	1.522e+03 +	**1.509e+03**	1.518e+03 +	**1.510e+03**	1.512e+03 +	**1.508e+03**	1.517e+03 +	**1.505e+03**	1.545e+03 ≈	**1.512e+03**	1.511e+03 +	**1.505e+03**
	std	1.826e+01	5.458e+00	8.038e+01	4.596e+00	1.228e+01	2.224e+00	2.429e+00	3.190e+00	2.281e+00	2.076e+00	5.312e+00	1.285e+00	4.394e+01	1.400e+01	4.129e+00	2.802e+00

Table 4 (continued)

Func.		DE_{rand}	DE_{rand}-AMP	DE_{cur}	DE_{cur}-AMP	DE_{best}	DE_{best}-AMP	$DE_{cur2best}$	$DE_{cur2best}$-AMP	$DE_{cur2pbest}$	$DE_{cur2pbest}$-AMP	JADE	JADE-AMP	GTDE	GTDE-AMP	CDE	CDE-AMP
f_{16}	mean	1.769e+03 +	**1.682e+03**	1.820e+03 +	**1.658e+03**	**1.624e+03** ≈	1.640e+03	1.682e+03 ≈	**1.675e+03**	1.679e+03 +	**1.634e+03**	1.672e+03 +	**1.616e+03**	1.693e+03 ≈	**1.657e+03**	**1.624e+03** ≈	1.650e+03
	std	3.400e+01	4.983e+01	7.144e+01	3.999e+01	1.908e+01	3.781e+01	4.609e+01	1.052e+02	2.763e+01	1.727e+01	4.402e+01	3.879e+01	6.083e+01	5.692e+01	1.516e+01	7.865e+01

f_1: Unimodal function; $f_3 - f_9$: Simple multimodal functions; $f_{10} - f_{19}$: Hybrid functions; $f_{20} - f_{30}$: Composite functions

4.2.2 Experimental results on CEC2022

Tables 8 and 9 summarize the experimental results and statistical analysis on CEC2022. Figure 6 presents the convergence curves of optimizers on representative functions of CEC2022 (i.e., f_1: Unimodal function; f_2 and f_5: Basic functions; f_6: Hybrid function; f_{11} and f_{12}: Composition functions.).

The experimental results and statistical analysis in Tables 8 and 9 demonstrate the competitive performance of the AMP selection mechanism when integrated into various DE-based optimizers. This remarkable performance showcases the excellent scalability of our proposal when used with different DE variants across distinct optimization problems. Additionally, the negative effects of the curse of dimensionality do not appear as the problem dimension increases from 10 to 20, suggesting that the proposed AMP selection mechanism remains effective and scalable in this range.

4.2.3 Sensitivity experiments on CEC2022

Considering the parameter k in the AMP selection mechanism plays an important role in choosing the number of competitors, this section presents the sensitivity experiments on CEC2022. As a well-known variant of DE, JADE-AMP (k=10) is employed as the base optimizer and $k \in \{4, 6, 8, 10, 12, 14\}$ are selected as candidates. Tables 10 and 11 summarize the results on CEC2022 benchmark functions.

The experimental results and statistical analysis in Tables 10 and 11 reveal that the statistical significance only exists in f_2 and f_{12} between JADE-AMP ($k = 4$) and JADE-AMP ($k = 10$) and f_1 and f_8 between JADE-AMP ($k = 8$) and JADE-AMP ($k = 10$) in 10-dimensional CEC2022 and f_5 between JADE-AMP ($k = 10$) with others in 20-dimensional CEC2022. In general, we can conclude that there are no statistically significant differences across different k settings for JADE-AMP. This consistency highlights the robustness and flexibility of the proposed AMP selection mechanism. Its ability to maintain stable performance regardless of parameter variation suggests that it adapts well to different optimization conditions, demonstrating exceptional scalability.

5 Discussion

This section discusses the advantages and disadvantages of the proposed AMP selection mechanism, and the potential topics for further developing the AMP selection mechanism are listed subsequently.

Table 5 The experimental results and statistical analysis on 10-dimensional CEC2017 benchmark functions (Continued)

Func.		DE_{rand}	DE_{rand}-AMP	DE_{cur}	DE_{cur}-AMP	DE_{best}	DE_{best}-AMP	$DE_{cur2best}$	$DE_{cur2best}$-AMP	$DE_{cur2pbest}$	$DE_{cur2pbest}$-AMP	JADE	JADE-AMP	GTDE	GTDE-AMP	CDE	CDE-AMP
f_{17}	mean	1.785e+03 +	**1.751e+03**	1.803e+03 +	**1.745e+03**	1.766e+03 ≈	**1.753e+03**	1.763e+03 +	**1.743e+03**	1.773e+03 +	**1.756e+03**	1.752e+03 +	**1.716e+03**	1.757e+03 +	**1.737e+03**	1.749e+03 +	**1.712e+03**
	std	1.024e+01	7.510e+00	1.735e+01	6.412e+00	4.733e+01	3.422e+01	9.185e+00	6.642e+00	8.489e+00	9.190e+00	9.929e+00	9.105e+00	3.437e+01	1.217e+01	5.832e+00	1.411e+01
f_{18}	mean	5.918e+03 +	**2.408e+03**	2.230e+04 +	**2.444e+03**	5.261e+03 +	**5.136e+03**	1.920e+03 +	**1.870e+03**	1.932e+03 +	**1.846e+03**	2.654e+03 +	**1.835e+03**	2.071e+03 +	**1.864e+03**	1.994e+03 +	**1.836e+03**
	std	2.704e+03	3.275e+02	9.154e+03	3.095e+02	9.789e+03	9.827e+03	3.002e+01	2.494e+01	6.162e+01	6.711e+00	1.259e+03	8.541e+00	2.318e+02	4.260e+01	1.029e+02	6.834e+00
f_{19}	mean	1.925e+03 +	**1.910e+03**	2.116e+03 +	**1.910e+03**	**1.908e+03** −	1.982e+03	1.908e+03 +	**1.905e+03**	1.906e+03 ≈	**1.906e+03**	1.910e+03 +	**1.903e+03**	1.912e+03 +	**1.906e+03**	1.906e+03 +	**1.902e+03**
	std	6.934e+00	1.287e+00	1.028e+02	1.837e+00	8.453e−01	2.303e+02	1.772e+00	6.413e−01	1.148e+00	4.899e−01	4.106e+00	9.360e−01	9.059e+00	7.554e+00	1.302e+00	1.113e+00
f_{20}	mean	2.076e+03 +	**2.042e+03**	2.116e+03 +	**2.039e+03**	2.029e+03 +	**2.015e+03**	2.058e+03 +	**2.019e+03**	2.077e+03 +	**2.056e+03**	2.046e+03 +	**2.013e+03**	2.033e+03 ≈	**2.023e+03**	2.031e+03 +	**2.005e+03**
	std	5.072e+00	9.020e+00	2.693e+01	5.588e+00	1.078e+01	1.127e+01	1.144e+01	5.357e+00	1.200e+01	1.289e+01	4.312e+00	9.457e+00	6.020e+00	1.654e+01	5.353e+00	7.845e+00
f_{21}	mean	**2.339e+03** ≈	2.340e+03	2.298e+03 ≈	**2.295e+03**	2.304e+03 ≈	**2.302e+03**	2.318e+03 +	**2.312e+03**	2.336e+03 +	**2.323e+03**	2.287e+03 +	**2.269e+03**	2.282e+03 +	**2.271e+03**	**2.251e+03** ≈	2.276e+03
	std	3.513e+01	2.137e+01	3.905e+01	6.533e+01	6.428e+01	6.423e+01	5.677e+01	5.402e+01	9.356e+00	3.940e+01	4.560e+01	5.614e+01	6.379e+01	5.717e+01	5.498e+01	4.814e+01
f_{22}	mean	2.334e+03 +	**2.315e+03**	2.504e+03 +	**2.315e+03**	2.309e+03 +	**2.308e+03**	2.309e+03 +	**2.308e+03**	2.307e+03 +	**2.306e+03**	2.305e+03 +	**2.300e+03**	2.307e+03 +	**2.302e+03**	2.303e+03 +	**2.302e+03**
	std	7.529e+00	1.359e+00	4.513e+01	9.546e−01	8.671e−01	9.554e−01	5.484e−01	1.220e+00	1.135e+00	7.499e−01	6.231e−01	1.687e−01	3.217e+00	1.511e+00	1.542e+00	7.761e−01
f_{23}	mean	**2.643e+03** ≈	2.644e+03	2.651e+03 +	**2.639e+03**	2.632e+03 ≈	**2.628e+03**	2.634e+03 ≈	**2.634e+03**	2.635e+03 ≈	**2.632e+03**	2.630e+03 +	**2.611e+03**	**2.624e+03** ≈	2.624e+03	2.625e+03 +	**2.609e+03**
	std	4.408e+00	6.914e+00	4.024e+00	4.950e+00	7.699e+00	9.900e+00	8.282e+00	9.859e+00	5.235e+00	5.310e+00	2.647e+00	6.933e+00	7.877e+00	6.934e+00	6.114e+00	5.789e+00
f_{24}	mean	2.776e+03 ≈	**2.772e+03**	**2.763e+03** ≈	2.773e+03	2.771e+03 ≈	**2.768e+03**	**2.768e+03** ≈	2.770e+03	**2.766e+03** ≈	2.768e+03	2.748e+03 +	**2.693e+03**	**2.734e+03** ≈	2.756e+03	**2.711e+03** ≈	2.715e+03
	std	7.226e+00	5.416e+00	3.150e+01	6.421e+00	5.173e+00	6.370e+00	6.392e+00	4.963e+00	4.847e+00	3.702e+00	4.195e+01	9.559e+01	6.322e+01	6.842e+00	9.165e+01	7.198e+01
f_{25}	mean	2.968e+03 +	**2.955e+03**	3.003e+03 +	**2.932e+03**	**2.921e+03** ≈	2.922e+03	2.922e+03 ≈	**2.917e+03**	**2.925e+03** ≈	2.927e+03	**2.926e+03** ≈	2.926e+03	2.942e+03 ≈	**2.938e+03**	**2.930e+03** −	2.942e+03
	std	3.594e+00	3.661e+00	1.082e+01	8.212e+00	2.152e+01	2.360e+01	2.060e+01	2.228e+01	2.074e+01	2.317e+01	2.254e+01	2.310e+01	1.447e+01	1.943e+01	2.100e+01	1.322e+01
f_{26}	mean	3.043e+03 +	**2.968e+03**	3.174e+03 +	**2.945e+03**	3.045e+03 +	**3.033e+03**	**2.919e+03** ≈	2.925e+03	2.902e+03 +	**2.900e+03**	**2.900e+03** ≈	2.909e+03	2.946e+03 ≈	**2.931e+03**	**2.906e+03** ≈	2.914e+03
	std	3.580e+01	8.710e+00	6.281e+01	7.562e+00	3.896e+02	3.459e+02	2.696e+01	3.923e+01	4.159e−01	3.157e−02	4.695e−01	1.886e+01	7.457e+01	7.460e+01	1.902e+01	3.812e+01
f_{27}	mean	3.096e+03 +	**3.093e+03**	3.097e+03 +	**3.093e+03**	3.091e+03 ≈	**3.091e+03**	3.090e+03 +	**3.090e+03**	3.091e+03 +	**3.090e+03**	**3.091e+03** ≈	3.092e+03	**3.092e+03** ≈	3.092e+03	**3.095e+03** ≈	3.097e+03
	std	9.060e−01	7.071e−01	1.771e+00	6.172e−01	1.024e+00	8.780e−01	8.468e−01	3.183e−01	6.411e−01	3.739e−01	6.636e−01	1.974e+00	2.626e+00	2.797e+00	2.107e+00	3.551e+00
f_{28}	mean	3.298e+03 ≈	**3.264e+03**	3.264e+03 +	**3.184e+03**	**3.249e+03** −	3.286e+03	**3.215e+03** ≈	3.237e+03	**3.227e+03** ≈	3.254e+03	3.220e+03 ≈	**3.199e+03**	3.321e+03 ≈	**3.291e+03**	**3.199e+03** ≈	3.233e+03
	std	4.070e+01	7.915e+01	2.412e+01	1.026e+01	1.020e+02	1.039e+02	9.267e+01	1.395e+02	1.222e+02	1.468e+02	1.288e+02	1.029e+02	1.100e+02	1.071e+02	1.343e+02	1.269e+02
f_{29}	mean	3.276e+03 +	**3.248e+03**	3.265e+03 +	**3.212e+03**	3.228e+03 +	**3.186e+03**	3.213e+03 +	**3.196e+03**	3.210e+03 +	**3.209e+03**	3.208e+03 +	**3.177e+03**	3.230e+03 ≈	**3.209e+03**	3.228e+03 +	**3.173e+03**
	std	2.282e+01	2.867e+01	4.349e+01	3.163e+01	3.199e+01	3.943e+01	2.155e+01	3.847e+01	2.723e+01	2.409e+01	2.360e+01	8.161e+00	6.923e+01	5.419e+01	2.570e+01	1.496e+01
f_{30}	mean	3.793e+05 +	**2.633e+05**	6.643e+05 +	**1.143e+05**	4.671e+05 +	**2.139e+05**	2.803e+05 +	**2.271e+05**	**3.525e+05** ≈	3.889e+05	3.028e+05 +	**1.696e+05**	**3.622e+05** ≈	3.764e+05	2.561e+05 +	**1.079e+05**
	std	2.926e+05	1.933e+05	2.039e+05	7.851e+04	4.434e+05	4.225e+05	5.066e+05	4.163e+05	4.013e+05	4.571e+05	4.215e+05	3.269e+05	3.782e+05	4.659e+05	4.755e+05	2.844e+05
+/≈/− summary:		23/6/0	–	27/2/0	–	15/11/3	–	19/10/0	–	21/8/0	–	25/4/0	–	16/13/0	–	21/7/1	–

Table 6 The experimental results and statistical analysis on 30-dimensional CEC2017 benchmark functions

Func.		DE_{rand}	DE_{rand}-AMP	DE_{cur}	DE_{cur}-AMP	DE_{best}	DE_{best}-AMP	$DE_{cur2best}$	$DE_{cur2best}$-AMP	$DE_{cur2pbest}$	$DE_{cur2pbest}$-AMP	JADE	JADE-AMP	GTDE	GTDE-AMP	CDE	CDE-AMP
f_1	mean	1.215e+10 +	**8.726e+09**	2.655e+10 +	**5.307e+09**	1.774e+08 +	**6.029e+06**	7.020e+07 +	**7.920e+06**	9.087e+06 +	**1.407e+05**	**1.255e+03** −	2.666e+03	1.890e+04 +	**9.534e+03**	**3.592e+03** −	2.794e+05
	std	1.267e+09	1.255e+09	2.267e+09	8.903e+08	9.078e+07	6.596e+06	3.974e+07	7.096e+06	6.209e+06	8.804e+04	1.932e+03	1.987e+03	1.432e+04	9.278e+03	3.314e+03	6.199e+05
f_3	mean	2.127e+05 ≈	**2.088e+05**	1.939e+05 ≈	**1.877e+05**	1.475e+05 ≈	**1.303e+05**	1.391e+05 ≈	**1.274e+05**	1.229e+05 +	**8.159e+04**	**3.165e+04** ≈	4.239e+04	9.586e+04 +	**5.117e+04**	6.035e+04 +	**2.572e+04**
	std	2.058e+04	3.712e+04	3.065e+04	2.693e+04	2.537e+04	1.690e+04	1.978e+04	2.319e+04	1.285e+04	8.417e+03	3.459e+04	2.203e+04	2.009e+04	1.851e+04	1.050e+04	5.578e+03
f_4	mean	1.309e+03 +	**8.491e+02**	2.915e+03 +	**7.664e+02**	5.183e+02 +	**4.964e+02**	5.055e+02 +	**4.927e+02**	4.905e+02 +	**4.873e+02**	5.057e+02 +	**4.911e+02**	5.004e+02 ≈	**4.922e+02**	5.152e+02 ≈	**5.146e+02**
	std	1.417e+02	8.097e+01	4.436e+02	5.174e+01	2.855e+01	1.562e+01	5.335e+00	7.599e+00	1.806e+00	6.017e−01	1.664e+01	1.986e+01	1.560e+01	2.886e+00	1.756e+01	1.912e+01
f_5	mean	7.997e+02 +	**7.908e+02**	8.690e+02 +	**7.754e+02**	7.505e+02 +	**7.321e+02**	7.362e+02 +	**7.285e+02**	7.108e+02 ≈	**7.066e+02**	6.072e+02 +	**5.295e+02**	6.759e+02 +	**5.877e+02**	6.818e+02 +	**5.429e+02**
	std	2.041e+01	1.372e+01	1.499e+01	1.660e+01	2.007e+01	1.137e+01	1.260e+01	9.544e+00	1.373e+01	1.072e+01	9.746e+00	6.002e+00	4.280e+01	2.252e+01	1.026e+01	1.401e+01
f_6	mean	6.468e+02 +	**6.403e+02**	6.728e+02 +	**6.337e+02**	6.109e+02 +	**6.034e+02**	6.092e+02 +	**6.028e+02**	6.027e+02 +	**6.002e+02**	6.000e+02 +	**6.000e+02**	6.032e+02 +	**6.017e+02**	6.001e+02 +	**6.000e+02**
	std	2.607e+00	2.676e+00	4.286e+00	5.098e+00	2.554e+00	1.337e+00	1.311e+00	7.994e−01	3.889e−01	3.026e−02	1.396e−02	7.987e−05	2.907e+00	9.941e−01	1.376e−01	4.781e−02
f_7	mean	1.561e+03 +	**1.422e+03**	2.056e+03 +	**1.218e+03**	1.001e+03 +	**9.710e+02**	9.980e+02 +	**9.812e+02**	9.608e+02 +	**9.485e+02**	8.459e+02 +	**7.668e+02**	9.344e+02 +	**8.318e+02**	9.302e+02 +	**7.754e+02**
	std	6.396e+01	4.017e+01	1.207e+02	5.885e+01	1.504e+01	1.852e+01	1.218e+01	7.965e+00	1.121e+01	1.497e+01	8.875e+00	1.092e+01	3.125e+01	2.380e+01	2.295e+01	1.233e+01
f_8	mean	1.111e+03 ≈	**1.102e+03**	1.142e+03 +	**1.080e+03**	1.059e+03 +	**1.031e+03**	1.041e+03 ≈	**1.037e+03**	1.022e+03 +	**1.002e+03**	9.076e+02 +	**8.305e+02**	9.721e+02 +	**8.976e+02**	9.801e+02 +	**8.441e+02**
	std	1.845e+01	8.522e+00	1.807e+01	1.534e+01	1.405e+01	2.107e+01	2.251e+01	1.543e+01	1.174e+01	1.445e+01	9.860e+00	7.182e+00	3.713e+01	2.439e+01	1.147e+01	8.729e+00
f_9	mean	1.051e+04 +	**7.957e+03**	1.748e+04 +	**5.577e+03**	2.784e+03 +	**1.121e+03**	1.778e+03 +	**1.013e+03**	9.975e+02 +	**9.006e+02**	9.019e+02 +	**9.000e+02**	1.977e+03 +	**1.967e+03**	9.182e+02 +	**9.167e+02**
	std	7.417e+02	1.349e+03	1.511e+03	9.997e+02	7.169e+02	1.248e+02	2.557e+02	6.179e+01	2.924e+01	1.639e−01	5.372e−02	2.124e+00	6.815e+02	8.811e+02	3.460e+01	9.755e+00
f_{10}	mean	8.605e+03 +	**8.415e+03**	8.652e+03 +	**7.761e+03**	8.453e+03 +	**6.894e+03**	8.423e+03 +	**7.878e+03**	8.603e+03 +	**8.366e+03**	6.365e+03 +	**4.259e+03**	7.677e+03 +	**4.589e+03**	8.403e+03 +	**5.883e+03**
	std	2.842e+02	3.560e+02	3.075e+02	6.126e+02	4.036e+02	1.621e+03	3.524e+02	1.181e+03	2.941e+02	4.557e+02	2.844e+02	7.347e+02	6.289e+02	6.728e+02	2.415e+02	2.172e+03
f_{11}	mean	2.776e+03 +	**2.277e+03**	4.022e+03 +	**1.877e+03**	1.397e+03 +	**1.307e+03**	1.353e+03 +	**1.298e+03**	1.280e+03 +	**1.240e+03**	1.203e+03 +	**1.190e+03**	1.345e+03 ≈	**1.334e+03**	1.219e+03 +	**1.189e+03**
	std	2.833e+02	1.655e+02	7.091e+02	1.527e+02	5.044e+01	3.067e+01	4.368e+01	4.550e+01	1.402e+01	2.635e+01	2.006e+01	4.481e+01	6.823e+01	7.017e+01	2.576e+01	3.809e+01
f_{12}	mean	7.594e+08 +	**1.597e+08**	9.365e+08 +	**1.276e+08**	2.212e+07 +	**2.795e+06**	2.189e+07 +	**2.536e+06**	1.983e+07 +	**1.511e+06**	**3.361e+04** −	7.386e+04	3.203e+06 +	**1.677e+05**	**3.051e+05** −	1.056e+06
	std	1.958e+08	6.168e+07	1.434e+08	6.406e+07	2.569e+07	3.976e+06	1.969e+07	1.408e+06	1.160e+07	9.523e+05	1.883e+04	6.924e+04	4.858e+06	2.048e+05	1.715e+05	3.851e+05
f_{13}	mean	4.650e+07 +	**5.221e+06**	6.578e+07 +	**3.810e+06**	3.406e+05 +	**4.619e+04**	3.499e+05 +	**2.062e+04**	5.227e+05 +	**2.345e+04**	**1.925e+03** −	1.305e+04	3.940e+04 +	**1.778e+04**	2.381e+04 +	**1.316e+04**
	std	1.586e+07	9.977e+06	1.138e+07	3.337e+06	5.911e+05	2.844e+04	2.556e+05	2.013e+04	3.203e+05	1.802e+04	4.223e+02	1.010e+04	2.628e+04	7.290e+03	1.062e+04	7.643e+03
f_{14}	mean	**6.846e+04** ≈	9.158e+04	1.755e+05 +	**2.375e+04**	**2.698e+03** −	2.901e+04	**1.941e+03** −	2.591e+03	**1.804e+03** −	2.400e+03	5.677e+03 ≈	**1.498e+03**	3.251e+03 +	**1.850e+03**	**4.972e+03** −	2.841e+04
	std	3.196e+04	3.656e+04	5.985e+04	1.191e+04	2.202e+03	8.173e+04	1.427e+02	5.845e+02	7.062e+01	6.425e+02	1.253e+04	2.873e+01	4.362e+03	1.618e+02	2.922e+03	4.072e+04
f_{15}	mean	8.775e+05 +	**4.317e+05**	2.909e+06 +	**2.538e+05**	1.830e+04 ≈	**1.820e+04**	**8.928e+03** −	3.858e+04	**7.756e+03** −	1.278e+04	**1.666e+03** −	2.143e+03	1.402e+04 +	**8.891e+03**	9.061e+03 +	**5.906e+03**
	std	2.303e+05	3.413e+05	8.115e+05	1.031e+05	1.397e+04	1.739e+04	2.431e+03	2.387e+04	3.190e+03	7.644e+03	7.135e+01	6.702e+02	1.506e+04	6.417e+03	4.439e+03	7.082e+03
f_{16}	mean	3.508e+03 ≈	**3.497e+03**	3.985e+03 +	**3.319e+03**	3.420e+03 +	**2.970e+03**	3.502e+03 +	**3.307e+03**	3.221e+03 ≈	**3.194e+03**	2.662e+03 +	**2.162e+03**	2.770e+03 +	**2.590e+03**	3.177e+03 +	**2.281e+03**
	std	1.312e+02	1.544e+02	1.534e+02	4.606e+02	3.101e+02	4.932e+02	1.895e+02	3.296e+02	1.557e+02	1.391e+02	2.257e+02	2.985e+02	2.400e+02	1.277e+02	1.663e+02	2.992e+02

Table 7 The experimental results and statistical analysis on 30-dimensional CEC2017 benchmark functions (Continued)

Func.		DE_{rand}	DE_{rand}-AMP	DE_{cur}	DE_{cur}-AMP	DE_{best}	DE_{best}-AMP	$DE_{cur2best}$	$DE_{cur2best}$-AMP	$DE_{cur2pbest}$	$DE_{cur2pbest}$-AMP	JADE	JADE-AMP	GTDE	GTDE-AMP	CDE	CDE-AMP
f_{17}	mean	2.659e+03 +	**2.617e+03**	2.709e+03 +	**2.099e+03**	2.246e+03 +	**2.159e+03**	2.382e+03 +	**2.150e+03**	2.411e+03 +	**2.264e+03**	2.096e+03 +	**1.835e+03**	2.192e+03 +	**2.008e+03**	2.022e+03 +	**1.870e+03**
	std	1.046e+02	1.473e+02	1.212e+02	1.060e+02	2.324e+02	2.135e+02	2.198e+02	2.571e+02	7.743e+01	1.170e+02	7.693e+01	1.253e+02	1.573e+02	2.114e+02	6.935e+01	9.241e+01
f_{18}	mean	1.000e+07 ≈	**8.503e+06**	8.648e+06 +	**1.144e+06**	1.388e+06 +	**5.151e+05**	1.730e+06 +	**8.435e+05**	**1.212e+06** ≈	2.201e+06	1.312e+06 +	**2.676e+03**	3.689e+05 +	**1.431e+05**	7.313e+05 +	**3.092e+05**
	std	3.220e+06	3.204e+06	3.114e+06	8.904e+05	6.147e+05	2.100e+05	7.290e+05	3.953e+05	4.453e+05	1.012e+06	3.931e+06	1.184e+03	3.141e+05	7.900e+04	4.088e+05	1.670e+05
f_{19}	mean	5.916e+06 +	**1.643e+06**	1.160e+07 +	**1.626e+06**	2.863e+04 ≈	**1.089e+04**	**2.929e+04** ≈	5.542e+04	**8.450e+03** ≈	8.612e+03	**4.001e+03** ≈	5.486e+03	**3.387e+03** ≈	4.801e+03	**1.134e+04** ≈	1.729e+04
	std	1.684e+06	1.751e+06	3.938e+06	5.106e+05	2.766e+04	7.048e+03	1.717e+04	6.621e+04	4.868e+03	6.529e+03	6.126e+03	5.415e+03	1.424e+03	1.734e+03	9.016e+03	8.894e+03
f_{20}	mean	2.917e+03 +	**2.634e+03**	2.959e+03 +	**2.424e+03**	2.586e+03 +	**2.311e+03**	2.719e+03 +	**2.332e+03**	2.760e+03 +	**2.532e+03**	2.496e+03 +	**2.372e+03**	**2.405e+03** −	2.594e+03	2.513e+03 +	**2.191e+03**
	std	1.391e+02	2.549e+02	8.527e+01	8.603e+01	2.576e+02	1.490e+02	1.564e+02	1.866e+02	1.289e+02	2.783e+02	1.991e+02	4.276e+02	8.255e+01	1.427e+02	1.283e+02	1.148e+02
f_{21}	mean	2.592e+03 ≈	**2.585e+03**	2.653e+03 +	**2.562e+03**	2.542e+03 +	**2.524e+03**	2.529e+03 ≈	**2.523e+03**	2.507e+03 ≈	**2.498e+03**	2.406e+03 +	**2.333e+03**	2.477e+03 +	**2.388e+03**	2.475e+03 +	**2.343e+03**
	std	1.593e+01	1.067e+01	1.501e+01	1.876e+01	1.384e+01	1.817e+01	1.135e+01	2.064e+01	7.342e+00	1.299e+01	8.939e+00	8.266e+00	3.803e+01	1.705e+01	1.671e+01	1.413e+01
f_{22}	mean	1.007e+04 ≈	**1.005e+04**	1.006e+04 +	**7.780e+03**	6.861e+03 +	**5.612e+03**	6.266e+03 +	**4.528e+03**	5.174e+03 +	**4.622e+03**	2.302e+03 +	**2.300e+03**	7.663e+03 +	**5.498e+03**	**2.301e+03** ≈	2.302e+03
	std	2.462e+02	1.574e+02	3.988e+02	3.059e+03	3.686e+03	3.452e+03	3.776e+03	3.387e+03	3.518e+03	3.548e+03	4.607e+00	2.424e−06	2.712e+03	1.700e+03	1.461e+00	2.201e+00
f_{23}	mean	2.918e+03 ≈	**2.913e+03**	2.991e+03 +	**2.901e+03**	2.886e+03 +	**2.873e+03**	2.873e+03 ≈	**2.868e+03**	2.862e+03 ≈	**2.845e+03**	2.766e+03 +	**2.682e+03**	2.808e+03 +	**2.764e+03**	2.788e+03 +	**2.704e+03**
	std	2.485e+01	1.253e+01	1.486e+01	1.292e+01	8.128e+00	1.205e+01	2.416e+01	1.728e+01	1.251e+01	1.111e+01	1.328e+01	1.424e+01	3.174e+01	2.780e+01	4.831e+01	1.281e+01
f_{24}	mean	3.072e+03 ≈	**3.060e+03**	3.150e+03 +	**3.060e+03**	3.044e+03 ≈	**3.039e+03**	3.036e+03 ≈	**3.031e+03**	3.025e+03 +	**3.014e+03**	2.942e+03 +	**2.850e+03**	2.986e+03 +	**2.942e+03**	2.991e+03 +	**2.872e+03**
	std	9.514e+00	1.471e+01	1.470e+01	1.312e+01	2.381e+01	1.223e+01	1.225e+01	1.541e+01	1.649e+01	1.335e+01	1.984e+01	1.137e+01	2.920e+01	2.695e+01	3.306e+01	1.362e+01
f_{25}	mean	3.972e+03 +	**3.512e+03**	5.254e+03 +	**3.198e+03**	2.922e+03 +	**2.890e+03**	2.907e+03 +	**2.889e+03**	2.889e+03 +	**2.887e+03**	**2.887e+03** ≈	2.890e+03	**2.889e+03** ≈	2.897e+03	**2.904e+03** ≈	2.924e+03
	std	1.695e+02	1.089e+02	2.070e+02	7.149e+01	1.785e+01	1.016e+00	4.952e+00	7.593e−01	4.027e−01	5.845e−02	1.202e+00	7.601e+00	9.974e−01	2.630e+01	1.961e+01	2.280e+01
f_{26}	mean	6.629e+03 +	**6.458e+03**	7.345e+03 +	**6.269e+03**	5.994e+03 ≈	**5.852e+03**	5.965e+03 +	**5.793e+03**	5.745e+03 +	**5.557e+03**	4.589e+03 +	**4.024e+03**	**4.584e+03** ≈	4.942e+03	4.490e+03 ≈	**4.252e+03**
	std	1.017e+02	1.751e+02	1.819e+02	1.937e+02	2.223e+02	1.796e+02	1.495e+02	1.468e+02	8.743e+01	1.257e+02	1.025e+02	1.844e+02	9.061e+02	1.816e+02	9.234e+02	1.554e+02
f_{27}	mean	3.277e+03 +	**3.248e+03**	3.304e+03 +	**3.242e+03**	**3.213e+03** ≈	3.223e+03	3.216e+03 +	**3.203e+03**	3.216e+03 +	**3.202e+03**	**3.209e+03** ≈	3.211e+03	**3.227e+03** ≈	3.244e+03	**3.225e+03** −	3.254e+03
	std	2.524e+01	7.737e+00	1.972e+01	1.095e+01	9.468e+00	2.280e+01	6.166e+00	8.710e+00	8.410e+00	9.167e+00	6.836e+00	8.580e+00	1.280e+01	2.801e+01	1.454e+01	1.910e+01
f_{28}	mean	4.076e+03 +	**3.629e+03**	4.550e+03 +	**3.492e+03**	3.327e+03 ≈	**3.316e+03**	3.289e+03 +	**3.240e+03**	3.244e+03 +	**3.228e+03**	3.598e+03 ≈	**3.586e+03**	**3.324e+03** ≈	3.369e+03	**3.249e+03** ≈	3.287e+03
	std	1.987e+02	7.004e+01	1.412e+02	4.064e+01	5.152e+01	5.303e+01	2.508e+01	2.585e+01	1.810e+01	2.299e+01	1.140e+03	1.107e+03	8.313e+01	1.112e+02	1.901e+01	1.548e+01
f_{29}	mean	4.936e+03 +	**4.592e+03**	4.793e+03 +	**4.151e+03**	4.171e+03 +	**3.699e+03**	4.356e+03 +	**3.814e+03**	4.223e+03 +	**3.945e+03**	3.633e+03 +	**3.445e+03**	3.783e+03 +	**3.669e+03**	3.867e+03 +	**3.573e+03**
	std	1.411e+02	1.899e+02	1.711e+02	2.913e+02	2.573e+02	1.611e+02	2.224e+02	1.796e+02	1.883e+02	2.310e+02	8.453e+01	5.068e+01	2.278e+02	2.147e+02	1.732e+02	1.030e+02
f_{30}	mean	5.641e+06 +	**2.694e+06**	1.237e+07 +	**1.554e+06**	1.295e+05 ≈	**1.033e+05**	1.823e+05 +	**5.647e+04**	1.058e+05 +	**6.429e+04**	2.822e+04 ≈	**6.628e+03**	3.108e+04 +	**1.412e+04**	5.982e+04 +	**3.180e+04**
	std	9.977e+05	8.012e+05	2.254e+06	6.847e+05	8.374e+04	1.523e+05	1.027e+05	3.087e+04	4.603e+04	5.888e+04	6.533e+04	4.787e+02	2.835e+04	4.944e+03	5.671e+04	3.211e+04
+/≈/− summary:		20/9/0	–	28/1/0	–	20/8/1	–	21/6/2	–	21/6/2	–	18/7/4	–	21/7/1	–	19/6/4	–

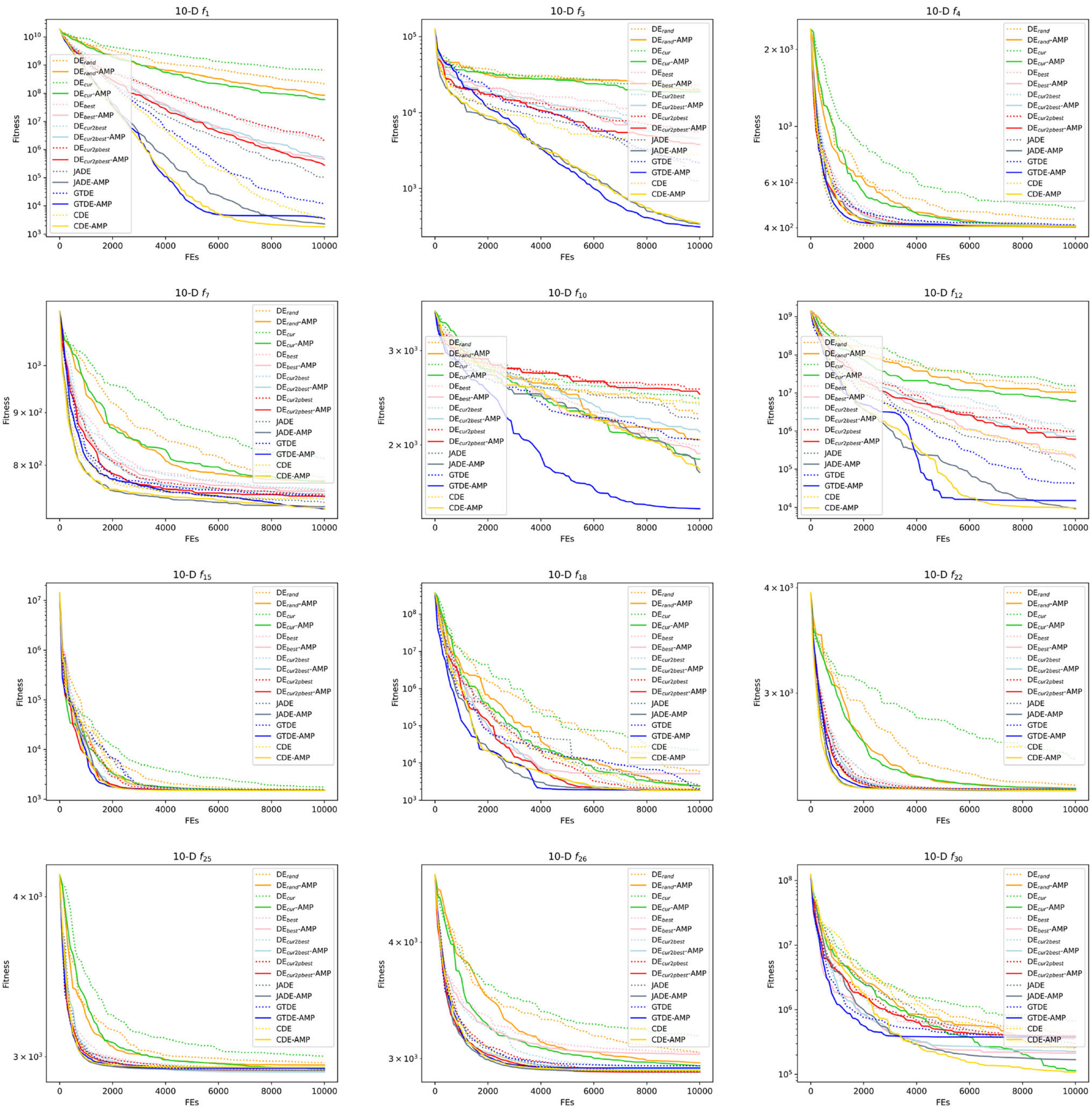


Fig. 4 Convergence curves of DE-based optimizers on representative functions of 10-dimensional CEC2017

Initially, the AMP selection mechanism introduces the competitive mechanism to the selection phase based on the Manhattan distance saved on the adjacent matrix, this probabilistic selection mechanism introduces randomness and allows the individuals who may be inferior but are far away from the current best solution to have a probability of being survival. This concept in the AMP selection mechanism design can maintain population diversity and enhance the explorative search behaviors during optimization. Although the distance metric is also considered in other selection mechanisms, the global adjacent matrix constructed by AMP can further capture the knowledge of the entire population and maintain the population diversity while accelerating the optimization convergence by ensuring the survival of elite individuals.

The second advantage is the implicit elitism strategy embedded in the proposed AMP selection mechanism. This strategy ensures that the current best individual will consistently win against all competitors and survive to the next generation. Conversely, the worst individual inevitably

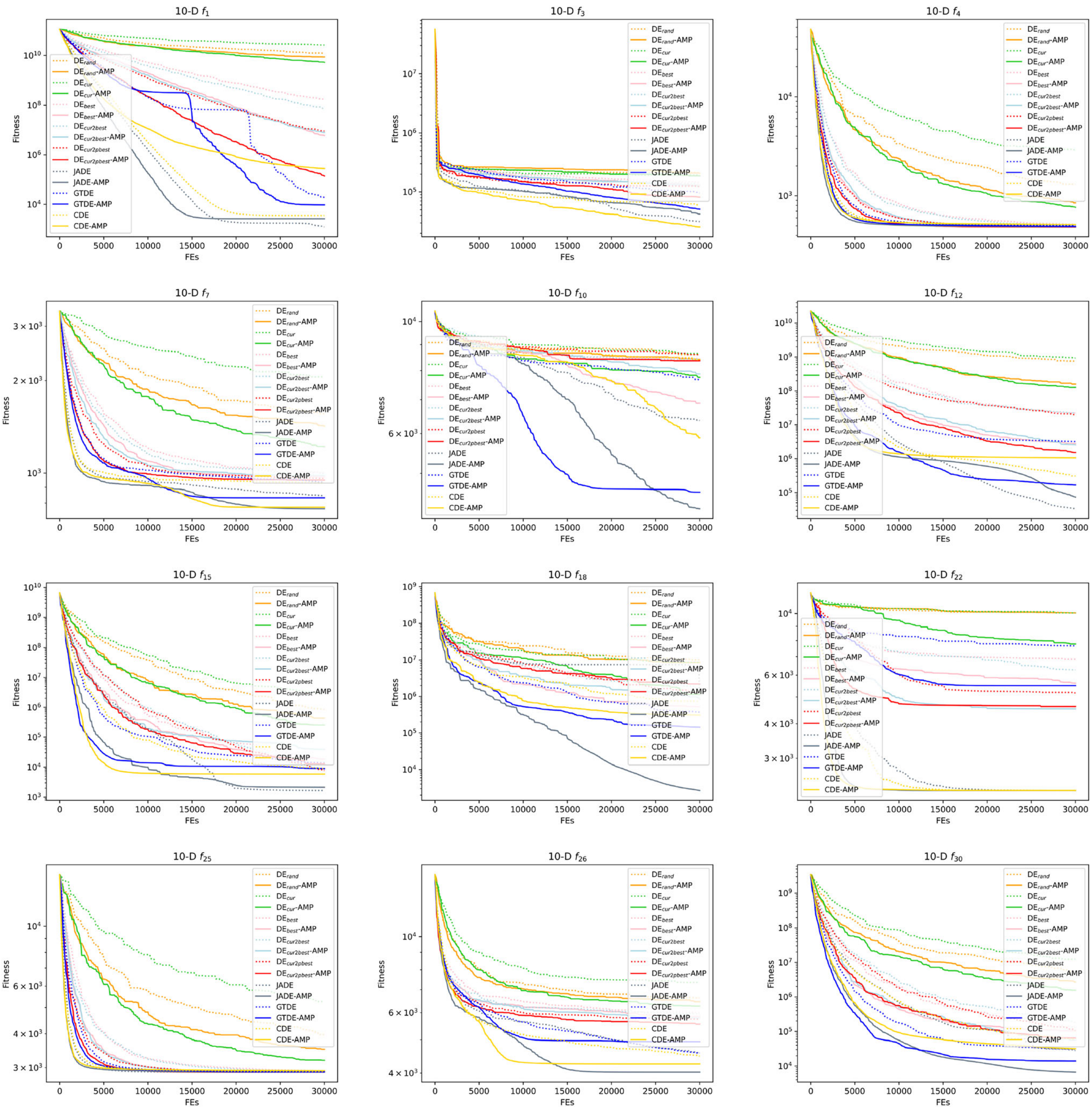


Fig. 5 Convergence curves of DE-based optimizers on representative functions of 30-dimensional CEC2017

loses the competition with other competitors and is discarded. This implicit elitism mechanism aligns with the principle of "survival of the fittest," which promotes the protection of high-quality solutions and accelerates the convergence process.

Additionally, since the AMP selection mechanism is proposed independently, it offers flexibility and adaptability, making it a promising candidate for extension to various evolutionary algorithms (EAs) and swarm intelligence (SI) approaches. The modular design of the AMP selection mechanism allows it to be seamlessly integrated into different optimization frameworks. This novel selection mechanism can strengthen optimization performance through the unique competitive mechanism and probabilistic selection characteristics.

However, there are still some shortcomings in the AMP selection mechanism. For instance, the AMP selection mechanism is computationally expensive, limiting its applicability to high-dimensional and large-scale

Table 8 The experimental results and statistical analysis on 10-D CEC2022 benchmark functions

Func.		DE_{rand}	DE_{rand}-AMP	DE_{cur}	DE_{cur}-AMP	DE_{best}	DE_{best}-AMP	$DE_{cur2best}$	$DE_{cur2best}$-AMP	$DE_{cur2pbest}$	$DE_{cur2pbest}$-AMP	JADE	JADE-AMP	GTDE	GTDE-AMP	CDE	CDE-AMP
f_1	Mean	4.342e+03 +	**3.288e+03**	6.049e+03 +	**2.097e+03**	6.227e+02 +	**3.074e+02**	5.466e+02 +	**3.135e+02**	4.030e+02 +	**3.012e+02**	3.004e+02 +	**3.000e+02**	3.026e+02 +	**3.000e+02**	3.000e+02 +	**3.000e+02**
	Std	1.269e+03	8.710e+02	1.053e+03	4.008e+02	7.895e+01	4.586e+00	9.681e+01	1.084e+01	3.522e+01	5.049e−01	5.078e−01	1.386e−07	1.883e+00	1.765e−08	1.728e−02	2.301e−06
f_2	Mean	4.258e+02 +	**4.141e+02**	4.669e+02 +	**4.117e+02**	4.092e+02 +	**4.056e+02**	4.067e+02 ≈	**4.062e+02**	4.081e+02 ≈	**4.080e+02**	4.077e+02 +	**4.026e+02**	4.168e+02 +	**4.091e+02**	4.082e+02 +	**4.030e+02**
	Std	6.177e+00	3.332e+00	1.628e+01	1.740e+00	3.518e+00	3.680e+00	2.719e+00	2.603e+00	1.531e+00	2.185e+00	3.169e+00	3.618e+00	2.461e+01	3.617e+00	2.139e+01	2.694e+00
f_3	Mean	6.000e+02 +	**6.000e+02**	6.000e+02 +	**6.000e+02**	6.000e+02 +	**6.000e+02**	6.000e+02 +	**6.000e+02**	6.000e+02 +	**6.000e+02**	6.000e+02 +	**6.000e+02**	6.000e+02 +	**6.000e+02**	6.000e+02 +	**6.000e+02**
	Std	1.501e−03	2.523e−04	8.721e−03	3.808e−04	2.986e−06	1.115e−07	3.044e−06	7.504e−08	5.230e−07	6.107e−09	6.665e−09	2.127e−13	4.191e−09	2.738e−13	1.569e−10	0.000e+00
f_4	Mean	8.012e+02 ≈	**8.011e+02**	8.011e+02 ≈	**8.010e+02**	8.010e+02 ≈	**8.010e+02**	8.011e+02 +	**8.007e+02**	8.010e+02 ≈	**8.010e+02**	8.007e+02 +	**8.005e+02**	8.005e+02 ≈	**8.004e+02**	8.007e+02 +	**8.002e+02**
	Std	2.405e−01	1.415e−01	1.812e−01	1.862e−01	2.740e−01	2.906e−01	1.305e−01	2.813e−01	1.945e−01	1.644e−01	3.078e−01	3.048e−01	2.218e−01	2.428e−01	1.478e−01	8.889e−02
f_5	Mean	9.007e+02 +	**9.002e+02**	9.026e+02 +	**9.003e+02**	9.000e+02 +	**9.000e+02**	9.000e+02 +	**9.000e+02**	9.000e+02 +	**9.000e+02**	9.000e+02 +	**9.000e+02**	**9.001e+02** ≈	9.002e+02	9.000e+02 +	**9.000e+02**
	Std	2.271e−01	6.517e−02	3.738e−01	1.004e−01	1.241e-02	1.526e−04	4.885e−03	1.312e−04	3.796e−04	4.440e−06	1.665e−05	1.923e−10	1.817e−01	5.339e−01	3.581e−02	2.686e−02
f_6	Mean	1.188e+05 +	**7.266e+04**	1.685e+05 +	**4.410e+04**	3.529e+04 +	**1.278e+04**	2.216e+04 +	**1.840e+04**	1.722e+04 +	**1.260e+04**	1.262e+04 +	**2.558e+03**	2.414e+04 +	**1.815e+04**	1.156e+04 +	**6.616e+03**
	Std	2.639e+04	2.450e+04	6.099e+04	1.178e+04	1.430e+04	6.559e+03	1.048e+04	8.989e+03	4.791e+03	5.570e+03	3.825e+03	7.742e+02	1.436e+04	1.221e+04	5.152e+03	2.776e+03
f_7	Mean	2.066e+03 +	**2.055e+03**	2.104e+03 +	**2.049e+03**	2.044e+03 +	**2.038e+03**	**2.045e+03** ≈	2.049e+03	2.042e+03 +	**2.037e+03**	2.039e+03 +	**2.026e+03**	2.061e+03 +	**2.045e+03**	2.034e+03 +	**2.026e+03**
	Std	1.318e+01	1.169e+01	2.840e+01	6.634e+00	8.975e+00	3.346e+01	6.974e+00	2.753e+01	4.943e+00	4.041e+00	6.739e+00	2.325e+00	4.516e+01	4.186e+01	3.183e+00	4.092e+00
f_8	Mean	2.250e+03 +	**2.240e+03**	2.306e+03 +	**2.235e+03**	2.249e+03 +	**2.224e+03**	2.229e+03 +	**2.227e+03**	2.229e+03 +	**2.225e+03**	2.228e+03 +	**2.218e+03**	2.308e+03 +	**2.261e+03**	2.223e+03 +	**2.218e+03**
	Std	7.564e+00	5.049e+00	2.677e+01	2.402e+00	3.870e+01	4.661e+00	1.242e+00	2.313e+00	2.486e+00	3.965e+00	3.501e+00	8.692e+00	1.634e+02	1.231e+02	5.641e+00	8.621e+00
f_9	Mean	2.658e+03 +	**2.619e+03**	2.698e+03 +	**2.556e+03**	2.663e+03 ≈	**2.661e+03**	2.626e+03 +	**2.588e+03**	2.660e+03 +	**2.587e+03**	2.410e+03 +	**2.373e+03**	**2.664e+03** ≈	2.669e+03	2.337e+03 +	**2.300e+03**
	Std	5.406e+01	8.671e+01	1.713e+01	1.360e+02	3.360e+00	3.623e+00	1.051e+02	1.439e+02	1.165e+00	1.436e+02	1.666e+02	1.451e+02	2.917e+00	7.748e+00	1.111e+02	4.330e−03
f_{10}	Mean	2.611e+03 +	**2.605e+03**	2.653e+03 +	**2.605e+03**	**2.631e+03** −	2.645e+03	2.600e+03 +	**2.599e+03**	**2.599e+03** ≈	2.614e+03	2.614e+03 +	**2.611e+03**	2.642e+03 ≈	**2.641e+03**	2.600e+03 ≈	**2.600e+03**
	Std	1.181e+00	8.223e−01	1.475e+01	1.327e+00	6.146e+01	7.032e+01	8.278e−01	7.632e−01	2.259e−01	4.454e+01	4.251e+01	3.658e+01	6.368e+01	6.042e+01	8.980e−01	1.241e+00
f_{11}	Mean	2.625e+03 +	**2.607e+03**	2.638e+03 +	**2.605e+03**	**2.778e+03** ≈	2.779e+03	**2.688e+03** −	2.775e+03	**2.600e+03** −	2.688e+03	2.600e+03 +	**2.600e+03**	**2.692e+03** −	2.791e+03	2.600e+03 +	**2.600e+03**
	Std	3.596e+00	1.931e+00	4.845e+00	9.319e−01	3.493e+02	3.485e+02	2.627e+02	3.504e+02	3.471e−02	2.628e+02	9.259e−03	6.224e−05	2.619e+02	3.424e+02	7.057e−04	2.951e−06
f_{12}	Mean	2.867e+03 +	**2.866e+03**	2.870e+03 +	**2.867e+03**	2.866e+03 ≈	**2.866e+03**	2.865e+03 ≈	**2.865e+03**	2.866e+03 +	**2.865e+03**	2.866e+03 +	**2.865e+03**	**2.867e+03** ≈	2.868e+03	2.869e+03 +	**2.867e+03**
	Std	5.036e−01	7.485e−01	6.181e−01	3.682e−01	6.419e−01	7.454e−01	7.959e−01	8.286e−01	9.346e−01	7.333e−01	5.740e−01	6.803e−01	1.845e+00	1.861e+00	5.200e+00	8.925e−01
+/≈/− summary:		11/1/0	–	11/1/0	–	7/4/1	–	8/3/1	–	8/3/1	–	12/0/0	–	6/5/1	–	11/1/0	–

f_1: Unimodal function; $f_2 - f_5$: Basic functions; $f_6 - f_8$: Hybrid functions; $f_9 - f_{12}$: Composition functions

Table 9 The experimental results and statistical analysis on 20-dimensional CEC2022 benchmark functions

Func.		DE_{rand}	DE_{rand}-AMP	DE_{cur}	DE_{cur}-AMP	DE_{best}	DE_{best}-AMP	$DE_{cur2best}$	$DE_{cur2best}$-AMP	$DE_{cur2pbest}$	$DE_{cur2pbest}$-AMP	JADE	JADE-AMP	GTDE	GTDE-AMP	CDE	CDE-AMP
f_1	Mean	2.464e+04 +	**1.839e+04**	3.109e+04 +	**1.287e+04**	4.553e+03 +	**1.128e+03**	3.404e+03 +	**9.945e+02**	1.543e+03 +	**3.543e+02**	3.000e+02 +	**3.000e+02**	3.234e+02 +	**3.000e+02**	3.000e+02 +	**3.000e+02**
	Std	5.278e+03	1.503e+03	5.268e+03	2.571e+03	1.526e+03	4.189e+02	9.753e+02	3.644e+02	2.767e+02	2.942e+01	3.288e−03	7.009e−09	2.402e+01	1.387e−06	1.456e−02	8.908e−04
f_2	Mean	6.170e+02 +	**5.299e+02**	9.016e+02 +	**5.085e+02**	4.494e+02 ≈	**4.487e+02**	4.498e+02 +	**4.479e+02**	4.488e+02 +	**4.487e+02**	4.552e+02 +	**4.487e+02**	4.488e+02 ≈	**4.463e+02**	4.708e+02 +	**4.624e+02**
	Std	3.387e+01	7.013e+00	1.281e+02	1.231e+01	1.976e+00	1.256e+00	7.445e−01	1.921e+00	1.070e+00	1.257e+00	9.503e+00	1.257e+00	1.670e+01	2.105e+01	1.683e+01	1.145e+01
f_3	Mean	6.001e+02 +	**6.000e+02**	6.003e+02 +	**6.000e+02**	6.000e+02 +	**6.000e+02**	6.000e+02 +	**6.000e+02**	6.000e+02 +	**6.000e+02**	6.000e+02 +	**6.000e+02**	6.000e+02 +	**6.000e+02**	6.000e+02 +	**6.000e+02**
	Std	9.404e−03	5.322e−03	3.357e−02	7.650e−03	1.328e−04	3.582e−06	3.891e−05	1.623e−06	1.011e−06	6.153e−09	1.922e−11	0.000e+00	1.888e−09	2.670e−12	3.938e−13	1.137e−13
f_4	Mean	**8.038e+02** ≈	8.039e+02	8.042e+02 ≈	**8.039e+02**	8.039e+02 +	**8.031e+02**	8.039e+02 ≈	**8.037e+02**	**8.036e+02** ≈	8.037e+02	8.021e+02 +	**8.018e+02**	8.014e+02 ≈	**8.010e+02**	8.035e+02 +	**8.006e+02**
	Std	5.310e−01	3.115e−01	2.703e−01	3.546e−01	5.103e−01	7.048e−01	4.361e−01	9.045e−01	3.142e−01	2.362e−01	3.242e−01	1.557e+00	6.021e−01	2.516e−01	2.500e−01	2.257e−01
f_5	Mean	9.084e+02 +	**9.042e+02**	9.135e+02 +	**9.029e+02**	9.007e+02 +	**9.003e+02**	9.004e+02 +	**9.000e+02**	9.000e+02 +	**9.000e+02**	9.001e+02 +	**9.000e+02**	**9.020e+02** ≈	9.042e+02	**9.002e+02** ≈	9.003e+02
	Std	8.200e−01	8.455e−01	2.026e+00	6.125e−01	2.423e−01	3.238e−01	1.661e−01	2.466e−03	5.340e−03	1.303e−05	1.624e−01	6.359e−08	2.144e+00	1.885e+00	2.073e−01	4.294e−01
f_6	Mean	2.317e+08 ≈	**2.140e+08**	1.994e+08 +	**4.444e+07**	2.826e+07 +	**7.518e+06**	3.318e+07 +	**1.114e+07**	4.903e+07 +	**2.256e+07**	**2.287e+04** −	3.724e+04	1.359e+05 +	**8.044e+04**	6.807e+05 +	**3.790e+04**
	Std	9.306e+07	9.422e+07	6.115e+07	1.889e+07	2.356e+07	1.202e+07	1.371e+07	8.646e+06	2.435e+07	1.034e+07	9.515e+03	9.480e+03	2.944e+04	2.737e+04	4.712e+05	1.071e+04
f_7	Mean	2.816e+03 +	**2.651e+03**	2.779e+03 +	**2.359e+03**	2.307e+03 +	**2.122e+03**	2.247e+03 +	**2.162e+03**	2.263e+03 +	**2.110e+03**	2.059e+03 +	**2.028e+03**	2.112e+03 +	**2.071e+03**	2.072e+03 +	**2.037e+03**
	Std	1.138e+02	1.676e+02	1.661e+02	6.643e+01	1.074e+02	6.347e+01	1.177e+02	9.083e+01	7.703e+01	1.327e+01	1.297e+01	4.058e+00	3.632e+01	6.869e+01	2.257e+01	7.511e+00
f_8	Mean	2.293e+06 +	**6.395e+05**	5.533e+04 +	**1.561e+04**	7.378e+03 +	**5.586e+03**	6.406e+03 +	**5.174e+03**	**7.580e+03** ≈	7.676e+03	2.273e+03 +	**2.238e+03**	3.577e+03 ≈	**3.524e+03**	3.372e+03 +	**2.814e+03**
	Std	4.136e+06	6.735e+05	4.917e+04	1.651e+04	2.238e+03	1.533e+03	1.525e+03	1.760e+03	5.797e+03	6.954e+03	1.305e+02	8.864e+00	1.498e+03	1.280e+03	3.995e+02	4.724e+02
f_9	Mean	2.740e+03 +	**2.683e+03**	2.853e+03 +	**2.669e+03**	2.641e+03 ≈	**2.641e+03**	2.642e+03 +	**2.636e+03**	2.638e+03 +	**2.636e+03**	**2.636e+03** ≈	2.636e+03	2.637e+03 +	**2.636e+03**	2.668e+03 +	**2.648e+03**
	Std	1.866e+01	7.607e+00	4.633e+01	8.761e+00	5.200e+00	6.029e+00	2.891e+00	2.295e−01	1.340e+00	4.337e−02	3.121e−01	2.568e−01	1.248e+00	1.133e+00	1.230e+01	4.960e+00
f_{10}	Mean	2.977e+03 +	**2.861e+03**	3.154e+03 +	**2.799e+03**	3.172e+03 +	**3.103e+03**	4.456e+03 +	**3.186e+03**	**2.812e+03** ≈	3.193e+03	2.810e+03 +	**2.794e+03**	**3.482e+03** ≈	3.951e+03	**2.802e+03** ≈	2.808e+03
	Std	1.788e+02	1.347e+02	9.132e+01	5.229e+00	1.031e+03	9.310e+02	2.020e+03	1.133e+03	9.439e+01	1.084e+03	8.615e+01	4.972e+01	1.274e+03	8.935e+02	5.990e+01	6.360e+01
f_{11}	Mean	2.666e+03 +	**2.634e+03**	2.727e+03 +	**2.625e+03**	2.822e+03 +	**2.758e+03**	2.605e+03 +	**2.600e+03**	2.600e+03 +	**2.600e+03**	2.600e+03 +	**2.600e+03**	**3.027e+03** ≈	3.051e+03	**2.600e+03** ≈	2.600e+03
	Std	3.315e+01	1.783e+01	2.296e+01	4.384e+00	4.743e+02	4.453e+02	1.079e+01	1.603e−01	3.190e−02	2.709e−03	1.771e−04	6.748e−06	6.481e+02	5.675e+02	4.680e−05	1.985e−01
f_{12}	Mean	2.950e+03 +	**2.943e+03**	2.970e+03 +	**2.946e+03**	**2.949e+03** ≈	2.955e+03	**2.947e+03** ≈	2.950e+03	2.942e+03 ≈	**2.938e+03**	**2.941e+03** ≈	**2.944e+03** ≈	2.957e+03 +	**2.951e+03**	**2.960e+03** ≈	2.966e+03
	Std	1.533e+00	2.550e+00	7.178e+00	3.255e+00	1.387e+01	1.502e+01	1.009e+01	1.197e+01	8.115e+00	2.409e+00	3.689e+00	7.513e+00	2.839e+01	8.206e+00	6.174e+00	1.323e+01
+/≈/− summary:		10/2/0	–	11/1/0	–	8/3/0	–	10/2/0	–	8/4/0	–	9/2/1	–	6/6/0	–	8/4/0	–

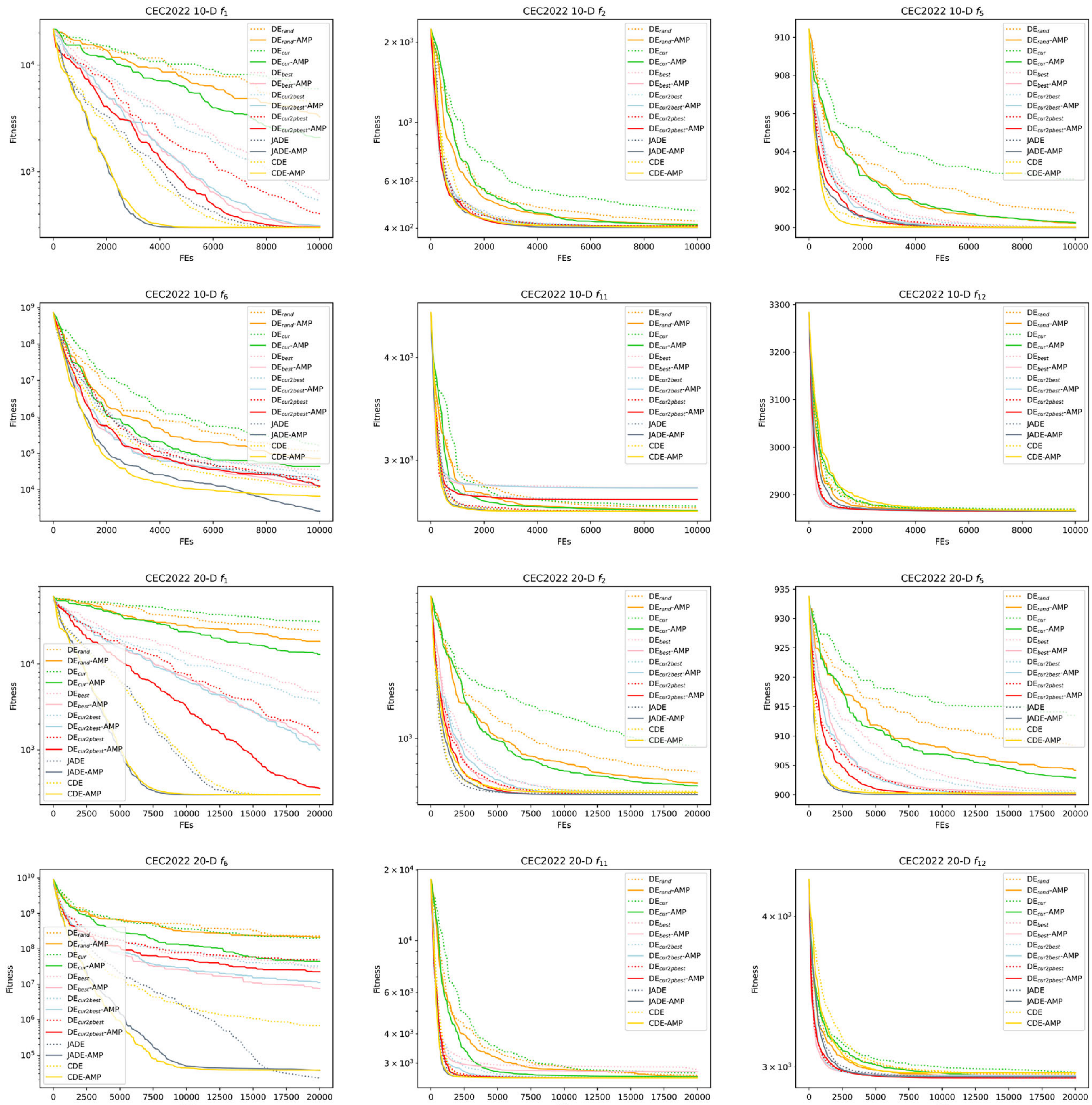


Fig. 6 Convergence curves of DE-based optimizers on representative functions of CEC2022

optimization problems. Therefore, we list some open topics for future research.

- **The integration with various optimization techniques**: The effectiveness and flexibility of the AMP selection mechanism is clear through the numerical experiments and analysis, and it is possible to extend this unique selection mechanism to various population-based optimization algorithms such as bio-inspired optimization approaches [42–44].
- **Reducing the necessary computational complexity**: The proposed AMP selection mechanism has a higher computational complexity and consumes significantly more computational resources than conventional selection methods, primarily due to the construction of the adjacency matrix. A pertinent question is whether we

Table 10 Results of sensitivity experiments on 10-dimensional CEC2022 benchmark functions

Func.		JADE-AMP (k=4)	JADE-AMP (k=6)	JADE-AMP (k=8)	JADE-AMP (k=12)	JADE-AMP (k=14)	JADE-AMP (k=10)
f_1	Mean	**3.000e+02** ≈	3.000e+02 ≈	3.000e+02 +	3.000e+02 ≈	3.000e+02 ≈	3.000e+02
	Std	8.075e−08	3.832e−05	4.339e−05	6.789e−06	5.379e−07	1.386e−07
f_2	Mean	4.062e+02 +	4.046e+02 ≈	4.048e+02 ≈	**4.025e+02** ≈	4.053e+02 ≈	4.026e+02
	Std	2.791e+00	3.979e+00	3.951e+00	3.614e+00	3.277e+00	3.618e+00
f_3	Mean	6.000e+02 ≈	6.000e+02 ≈	6.000e+02 ≈	6.000e+02 ≈	6.000e+02 ≈	**6.000e+02**
	Std	4.115e−13	9.102e−13	2.795e−12	3.872e−13	3.155e−13	2.127e−13
f_4	Mean	8.006e+02 ≈	8.006e+02 ≈	8.006e+02 ≈	8.007e+02 ≈	8.007e+02 ≈	**8.005e+02**
	Std	1.918e−01	2.685e−01	2.897e−01	3.916e−01	3.491e−01	3.048e−01
f_5	Mean	9.000e+02 ≈	9.000e+02 ≈	9.000e+02 ≈	9.000e+02 ≈	9.000e+02 ≈	**9.000e+02**
	Std	5.673e−10	8.087e−10	8.241e−10	2.246e−07	1.374e−09	1.923e−10
f_6	Mean	3.690e+03 ≈	3.147e+03 ≈	5.380e+03 ≈	3.257e+03 ≈	2.875e+03 ≈	**2.558e+03**
	Std	3.310e+03	1.256e+03	5.394e+03	2.855e+03	8.937e+02	7.742e+02
f_7	Mean	2.026e+03 ≈	2.026e+03 ≈	2.026e+03 ≈	**2.024e+03** ≈	2.026e+03 ≈	2.026e+03
	Std	1.934e+00	3.073e+00	6.192e+00	6.009e+00	2.474e+00	2.325e+00
f_8	Mean	2.218e+03 ≈	2.224e+03 ≈	**2.216e+03** −	2.220e+03 ≈	2.217e+03 ≈	2.218e+03
	Std	7.750e+00	2.785e+00	8.784e+00	4.901e+00	6.773e+00	8.692e+00
f_9	Mean	2.372e+03 ≈	2.337e+03 ≈	2.336e+03 ≈	2.300e+03 ≈	2.300e+03 ≈	2.373e+03
	Std	1.437e+02	1.096e+02	1.077e+02	1.118e−03	4.691e−02	1.451e+02
f_{10}	Mean	**2.599e+03** ≈	2.610e+03 ≈	2.611e+03 ≈	2.611e+03 ≈	2.630e+03 ≈	2.611e+03
	Std	3.605e−01	3.471e+01	3.583e+01	3.715e+01	9.329e+01	3.658e+01
f_{11}	Mean	2.600e+03 ≈	2.600e+03 ≈	**2.600e+03** ≈	2.600e+03 ≈	2.600e+03 ≈	2.600e+03
	Std	3.439e−04	1.616e−05	5.903e−05	3.474e−03	5.290e−05	6.224e−05
f_{12}	Mean	2.866e+03 +	2.866e+03 ≈	2.866e+03 ≈	2.865e+03 ≈	2.865e+03 ≈	2.865e+03
	Std	7.471e−01	6.911e−01	7.771e−01	7.539e−01	7.303e−01	6.803e−01
+/≈/− summary:		2/10/0	0/12/0	1/10/1	0/12/0	0/12/0	–

can avoid constructing the adjacency matrix and still compute the selection probabilities for competitors. Alternatively, a random selection approach may perform competitively and require fewer computational resources.

6 Conclusion

This paper proposes a novel selection scheme for DE named AMP selection mechanism. The proposed AMP selection mechanism introduces the concept of competition to the selection phase and constructs the adjacent matrix using the Manhattan distance between each pair of individuals. The Manhattan distance is employed to calculate the probability of being selected as the competitor, and the winner counts within the competition are adopted as the metric to rank the survival individuals. To evaluate the performance of our proposal, we integrate it with eight variants of DE and conduct numerical experiments on CEC2017 and CEC2022 benchmark functions. The experimental results and statistical analysis confirm the effectiveness and competitiveness of our proposal.

At the end of this paper, we discuss the advantages and disadvantages of the proposed AMP selection mechanism. Finally, we will continue to investigate the performance of the AMP selection mechanism and integrate it into optimization algorithms to further address various optimization tasks.

Table 11 Results of sensitivity experiments on 20-dimensional CEC2022 benchmark functions

Func.		JADE-AMP (k=4)	JADE-AMP (k=6)	JADE-AMP (k=8)	JADE-AMP (k=12)	JADE-AMP (k=14)	JADE-AMP (k=10)
f_1	Mean	3.000e+02 ≈	3.000e+02 ≈	**3.000e+02** ≈	3.000e+02 ≈	3.000e+02 ≈	3.000e+02
	Std	6.422e−05	6.221e−09	7.778e−09	7.678e−05	9.135e−05	7.009e−09
f_2	Mean	4.499e+02 ≈	4.504e+02 ≈	4.504e+02 ≈	4.508e+02 ≈	4.494e+02 ≈	**4.487e+02**
	Std	2.709e+00	6.597e+00	2.649e+00	4.227e+00	3.878e+00	1.257e+00
f_3	Mean	**6.000e+02** ≈	**6.000e+02** ≈	**6.000e+02** ≈	**6.000e+02** ≈	**6.000e+02** ≈	**6.000e+02**
	Std	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00
f_4	Mean	8.023e+02 ≈	8.025e+02 ≈	**8.014e+02** ≈	8.025e+02 ≈	8.023e+02 ≈	8.018e+02
	Std	1.323e+00	1.309e+00	1.225e+00	1.217e+00	1.309e+00	1.557e+00
f_5	Mean	9.000e+02 +	9.000e+02 +	9.000e+02 +	9.000e+02 +	9.000e+02 +	**9.000e+02**
	Std	3.581e−02	5.733e−02	2.686e−02	1.483e−06	3.581e−02	6.359e−08
f_6	Mean	4.401e+04 ≈	4.771e+04 ≈	**3.390e+04** ≈	3.791e+04 ≈	4.471e+04 ≈	3.724e+04
	Std	1.435e+04	9.729e+03	9.610e+03	1.610e+04	1.677e+04	9.480e+03
f_7	Mean	2.030e+03 ≈	2.030e+03 ≈	**2.028e+03** ≈	2.028e+03 ≈	2.029e+03 ≈	2.028e+03
	Std	3.866e+00	2.959e+00	1.754e+00	3.221e+00	1.959e+00	4.058e+00
f_8	Mean	2.295e+03 ≈	2.268e+03 ≈	2.275e+03 ≈	2.295e+03 ≈	2.293e+03 ≈	**2.238e+03**
	Std	1.429e+02	4.642e+01	6.090e+01	1.733e+02	1.002e+02	8.864e+00
f_9	Mean	2.637e+03 ≈	2.637e+03 ≈	2.636e+03 ≈	2.637e+03 ≈	**2.636e+03** ≈	2.636e+03
	Std	2.215e−01	2.513e−01	6.536e−01	1.388e−01	2.167e−01	2.568e−01
f_{10}	Mean	2.786e+03 ≈	2.785e+03 ≈	**2.775e+03** ≈	2.875e+03 ≈	2.798e+03 ≈	2.794e+03
	Std	5.363e+01	4.838e+01	4.067e+01	1.606e+02	5.481e+01	4.972e+01
f_{11}	Mean	2.600e+03 ≈	2.600e+03 ≈	2.600e+03 ≈	2.600e+03 ≈	2.600e+03 ≈	**2.600e+03**
	Std	1.847e−04	5.236e−06	8.701e−06	1.370e−06	1.544e−04	6.748e−06
f_{12}	Mean	**2.940e+03** ≈	2.945e+03 ≈	2.942e+03 ≈	2.947e+03 ≈	2.941e+03 ≈	2.944e+03
	Std	2.071e+00	6.949e+00	3.292e+00	1.066e+01	3.141e+00	7.513e+00
+/≈/− summary:		1/11/0	1/11/0	1/11/0	1/11/0	1/11/0	–

Author contributions **Rui Zhong**: Conceptualization, Methodology, Data curation, Software, Writing - original draft, Writing - review & editing, and Funding acquisition. **Shilong Zhang**: Investigation, Formal Analysis, and Writing - review & editing. **Yujun Zhang**: Methodology, Resources, and Writing - review & editing. **Jun Yu**: Supervision, Writing - review & editing, and Project administration.

Funding This work was supported by JST SPRING Grant Number JPMJSP2119, which is the funding of this research.

Data availability The source code of this research can be downloaded from https://github.com/RuiZhong961230/AMP.

Declarations

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

References

1. Wang, G.-G., Gao, D., Pedrycz, W.: Solving multi objective fuzzy job-shop scheduling problem by a hybrid adaptive differential evolution algorithm. IEEE Trans. Ind. Inf. **18**(12), 8519–8528 (2022). https://doi.org/10.1109/TII.2022.3165636
2. Song, Y., Zhao, G., Zhang, B., Chen, H., Deng, W., Deng, W.: An enhanced distributed differential evolution algorithm for portfolio optimization problems. Eng. Appl. Artif. Intell. **121**, 106004 (2023). https://doi.org/10.1016/j.engappai.2023.106004
3. Zhang, Y., Li, S., Wang, Y., Yan, Y., Zhao, J., Gao, Z.: Self-adaptive enhanced learning differential evolution with surprisingly efficient decomposition approach for parameter identification of photovoltaic models. Energy Convers. Manag. **308**, 118387 (2024). https://doi.org/10.1016/j.enconman.2024.118387
4. Zhong, R., Yu, J.: Dea2h2: differential evolution architecture based adaptive hyper-heuristic algorithm for continuous optimization. Cluster Comput. (2024). https://doi.org/10.1007/s10586-024-04587-0
5. Storn, R., Price, K.: Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J. Global Optim. **11**, 341–359 (1997). https://doi.org/10.1023/A:1008202821328
6. Li, Y., Han, T., Zhou, H., Tang, S., Zhao, H.: A novel adaptive l-shade algorithm and its application in uav swarm resource configuration problem. Inf. Sci. **606**, 350–367 (2022). https://doi.org/10.1016/j.ins.2022.05.058
7. Gu, Q., Li, S., Gong, W., Ning, B., Hu, C., Liao, Z.: L-shade with parameter decomposition for photovoltaic modules parameter identification under different temperature and irradiance. Appl.

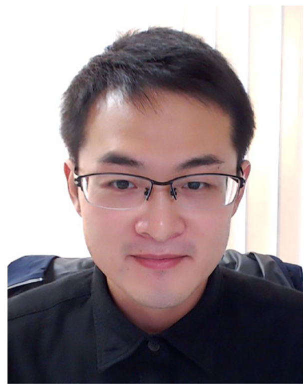
Soft Comput. **143**, 110386 (2023). https://doi.org/10.1016/j.asoc.2023.110386
8. Zhang, J., Chen, D., Yang, Q., Wang, Y., Liu, D., Jeon, S.-W., Zhang, J.: Proximity ranking-based multimodal differential evolution. Swarm Evol. Comput. **78**, 101277 (2023). https://doi.org/10.1016/j.swevo.2023.101277
9. Li, C., Sun, G., Deng, L., Qiao, L., Yang, G.: A population state evaluation-based improvement framework for differential evolution. Inf. Sci. **629**, 15–38 (2023). https://doi.org/10.1016/j.ins.2023.01.120
10. Meng, Z.: Dimension improvements based adaptation of control parameters in differential evolution: a fitness-value-independent approach. Expert Syst. Appl. **223**, 119848 (2023). https://doi.org/10.1016/j.eswa.2023.119848
11. Cai, Z., Yang, X., Zhou, M., Zhan, Z.-H., Gao, S.: Toward explicit control between exploration and exploitation in evolutionary algorithms: a case study of differential evolution. Inf. Sci. **649**, 119656 (2023). https://doi.org/10.1016/j.ins.2023.119656
12. Wang, K., Gao, S., Zhou, M., Zhan, Z.-H., Cheng, J.: Fractional order differential evolution. IEEE Trans. Evol. Comput. (2024). https://doi.org/10.1109/TEVC.2024.3382047
13. Yang, J., Wang, K., Wang, Y., Wang, J., Lei, Z., Gao, S.: Dynamic population structures-based differential evolution algorithm. IEEE Trans. Emerg. Top. Comput. Intell. **8**(3), 2493–2505 (2024). https://doi.org/10.1109/TETCI.2024.3367809
14. Sui, Q., Yu, Y., Wang, K., Zhong, L., Lei, Z., Gao, S.: Best-worst individuals driven multiple-layered differential evolution. Inf. Sci. **655**, 119889 (2024). https://doi.org/10.1016/j.ins.2023.119889
15. Liu, D., He, H., Yang, Q., Wang, Y., Jeon, S.-W., Zhang, J.: Function value ranking aware differential evolution for global numerical optimization. Swarm Evol. Comput. **78**, 101282 (2023). https://doi.org/10.1016/j.swevo.2023.101282
16. Yang, X., Wang, R., Zhao, D., Yu, F., Heidari, A.A., Xu, Z., Chen, H., Algarni, A.D., Elmannai, H., Xu, S.: Multi-level threshold segmentation framework for breast cancer images using enhanced differential evolution. Biomed. Signal Process. Control **80**, 104373 (2023). https://doi.org/10.1016/j.bspc.2022.104373
17. Song, Y., Cai, X., Zhou, X., Zhang, B., Chen, H., Li, Y., Deng, W., Deng, W.: Dynamic hybrid mechanism-based differential evolution algorithm and its application. Expert Syst. Appl. **213**, 118834 (2023). https://doi.org/10.1016/j.eswa.2022.118834
18. Li, Y., Han, T., Tang, S., Huang, C., Zhou, H., Wang, Y.: An improved differential evolution by hybridizing with estimation-of-distribution algorithm. Inf. Sci. **619**, 439–456 (2023). https://doi.org/10.1016/j.ins.2022.11.029
19. Yu, X., Jiang, N., Wang, X., Li, M.: A hybrid algorithm based on grey wolf optimizer and differential evolution for uav path planning. Expert Syst. Appl. **215**, 119327 (2023). https://doi.org/10.1016/j.eswa.2022.119327
20. Gu, Q., Li, S., Liao, Z.: Solving nonlinear equation systems based on evolutionary multitasking with neighborhood-based speciation differential evolution. Expert Syst. Appl. **238**, 122025 (2024). https://doi.org/10.1016/j.eswa.2023.122025
21. Zhang, Z., Yu, Q., Yang, H., Li, J., Cheng, J., Gao, S.: Triple-layered chaotic differential evolution algorithm for layout optimization of offshore wave energy converters. Expert Syst. Appl. **239**, 122439 (2024). https://doi.org/10.1016/j.eswa.2023.122439
22. Pant, M., Zaheer, H., Garcia-Hernandez, L., Abraham, A.: Differential evolution: a review of more than two decades of research. Eng. Appl. Artif. Intell. **90**, 103479 (2020). https://doi.org/10.1016/j.engappai.2020.103479
23. Ahmad, M.F., Isa, N.A.M., Lim, W.H., Ang, K.M.: Differential evolution: a recent review based on state-of-the-art works. Alex. Eng. J. **61**(5), 3831–3872 (2022). https://doi.org/10.1016/j.aej.2021.09.013
24. Chakraborty, S., Saha, A.K., Ezugwu, A.E., Agushaka, J.O., Zitar, R.A., Abualigah, L.: Differential evolution and its applications in image processing problems: a comprehensive review. Arch. Comput. Methods Eng. **30**(2), 985–1040 (2023). https://doi.org/10.1007/s11831-022-09825-5
25. Nadimi-Shahraki, M.H., Taghian, S., Mirjalili, S., Faris, H.: Mtde: an effective multi-trial vector-based differential evolution algorithm and its applications for engineering design problems. Appl. Soft Comput. **97**, 106761 (2020). https://doi.org/10.1016/j.asoc.2020.106761
26. Li, J.-Y., Du, K.-J., Zhan, Z.-H., Wang, H., Zhang, J.: Distributed differential evolution with adaptive resource allocation. IEEE Trans. Cybern. **53**(5), 2791–2804 (2023). https://doi.org/10.1109/TCYB.2022.3153964
27. Li, S., Gong, W., Wang, L., Yan, X., Hu, C.: Optimal power flow by means of improved adaptive differential evolution. Energy **198**, 117314 (2020). https://doi.org/10.1016/j.energy.2020.117314
28. Xue, Y., Tong, Y., Neri, F.: An ensemble of differential evolution and Adam for training feed-forward neural networks. Inf. Sci. **608**, 453–471 (2022). https://doi.org/10.1016/j.ins.2022.06.036
29. Das, S., Suganthan, P.N.: Differential evolution: a survey of the state-of-the-art. IEEE Trans. Evol. Comput. **15**(1), 4–31 (2011). https://doi.org/10.1109/TEVC.2010.2059031
30. Padhye, N., Bhardawaj, P., Deb, K.: Improving differential evolution through a unified approach. J. Glob. Optim. (2013). https://doi.org/10.1007/s10898-012-9897-0
31. Ho-Huu, V., Truong, T., Le, L., Vo, D.-T.: An improved differential evolution based on roulette wheel selection for shape and size optimization of truss structures with frequency constraints. Neural Comput. Appl. **29**, 167–185 (2018). https://doi.org/10.1007/s00521-016-2426-1
32. Tagawa, K.: Survival selection methods for the differential evolution based on continuous generation model. In: 2009 International Symposium on Autonomous Decentralized Systems, pp. 1–6 (2009). https://doi.org/10.1109/ISADS.2009.5207345
33. Ghosh, A., Das, S., Mallipeddi, R., Das, A.K., Dash, S.S.: A modified differential evolution with distance-based selection for continuous optimization in presence of noise. IEEE Access **5**, 26944–26964 (2017). https://doi.org/10.1109/ACCESS.2017.2773825
34. Tian, M., Gao, X., Dai, C.: Differential evolution with improved individual-based parameter setting and selection strategy. Appl. Soft Comput. **56**, 286–297 (2017). https://doi.org/10.1016/j.asoc.2017.03.010
35. Zeng, Z., Zhang, M., Chen, T., Hong, Z.: A new selection operator for differential evolution algorithm. Knowl. Based Syst. **226**, 107150 (2021). https://doi.org/10.1016/j.knosys.2021.107150
36. Yaguchi, K., Tamura, K., Yasuda, K., Ishigame, A.: Basic study of proximate optimality principle based combinatorial optimization method. In: 2011 IEEE International Conference on Systems, Man, and Cybernetics, pp. 1753–1758 (2011). https://doi.org/10.1109/ICSMC.2011.6083925
37. Zhong, R., Yu, J.: Hybrid remora crayfish optimization for engineering and wireless sensor network coverage optimization. Cluster Comput. (2024). https://doi.org/10.1007/s10586-024-04593-2
38. Thieu, N.V.: Opfunu: an open-source python library for optimization benchmark functions. J. Open Res. Softw. (2024). https://doi.org/10.5334/jors.508
39. Zhang, J., Sanderson, A.C.: Jade: adaptive differential evolution with optional external archive. IEEE Trans. Evol. Comput. **13**(5), 945–958 (2009). https://doi.org/10.1109/TEVC.2009.2014613
40. Wang, Z.-J., Jian, J.-R., Zhan, Z.-H., Li, Y., Kwong, S., Zhang, J.: Gene targeting differential evolution: a simple and efficient

method for large-scale optimization. IEEE Trans. Evol. Comput. **27**(4), 964–979 (2023). https://doi.org/10.1109/TEVC.2022.3185665
41. Zhong, R., Cao, Y., Zhang, E., Munetomo, M.: Introducing competitive mechanism to differential evolution for numerical optimization (2024). https://arxiv.org/abs/2406.05436
42. Zhong, R., Yu, J., Zhang, C., Munetomo, M.: Srime: a strengthened rime with Latin hypercube sampling and embedded distance-based selection for engineering optimization problems. Neural Comput. Appl. **36**, 6721–6740 (2024). https://doi.org/10.1007/s00521-024-09424-4
43. Zhang, Y.-J., Wang, Y.-F., Yan, Y.-X., Zhao, J., Gao, Z.-M.: Self-adaptive hybrid mutation slime mould algorithm: case studies on uav path planning, engineering problems, photovoltaic models and infinite impulse response. Alex. Eng. J. **98**, 364–389 (2024). https://doi.org/10.1016/j.aej.2024.04.075
44. Xu, Y., Zhong, R., Zhang, C., Yu, J.: Multiplayer battle game-inspired optimizer for complex optimization problems. Cluster Comput. (2024). https://doi.org/10.1007/s10586-024-04448-w

Rui Zhong received the B.Eng. degree from Huazhong Agricultural University, China in 2019, the M.Eng. degree from Kyushu University, Japan in 2022, and a Ph.D. at Hokkaido University, Japan in 2024. He is now a Specifically Appointed Assistant Professor at Information Initiative Center, Hokkaido University. His research interests include Evolutionary Computation, Large-scale Global Optimization, and Meta-/Hyper-heuristics. His profile in google scholar can be found at https://scholar.google.com/citations?hl=en&user=xd1vrwIAAAAJ&view_op=list_works&sortby=pubdate.

Shilong Zhang received a Bachelor's degree in Mathematics and Applied Mathematics from Huaqiao University, China. He is currently pursuing a Master's degree at Niigata University, Japan. His research focuses on evolutionary computation, deep reinforcement learning, and optimization techniques for neural networks.

Yujun Zhang received his Bachelor of Engineering degree from Jingchu University of Technology in 2023. His research interests include swarm intelligence and evolutionary computation.



Jun Yu received a Bachelor degree from Northeastern University, China in 2014, and a Master degree and a doctorate from Kyushu University, Japan in 2017 and 2019, respectively. He is currently an Assistant Professor at Niigata University, Japan. His research interests include evolutionary computation, artificial neural networks, and machine learning.