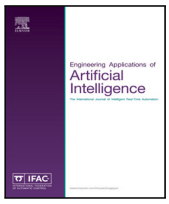
Engineering Applications of Artificial Intelligence 181 (2026) 115602

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

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



Research paper

Adaptive dual-module cooperation and competition differential evolution

Ying Xia Wu [a,1], Yu Hong Wang [a,1], Sheng Xin Zhang [a,*], Li Ming Zheng [a], Shao Yong Zheng [b]

[a] College of Information Science and Technology, Jinan University, Guangzhou, China
[b] School of Electronics and Information Technology and National and Provincial Joint Engineering Laboratory of IoT IC and System Application Technologies, Sun Yat-sen University, Guangzhou, China

ARTICLE INFO

Keywords:
Differential evolution
Cooperation and competition
Strategy adaptation
Numerical optimization

ABSTRACT

Balancing exploration and exploitation is a critical issue for improving the global search performance of differential evolution (DE). Existing strategies can generally be categorized into cooperation-driven approaches, which emphasize information sharing, and competition-driven approaches, which rely on historical experience for resource allocation. Despite notable progress, effectively adapting these strategies to diverse problem characteristics remains challenging, as balancing often entails trade-offs that may lead to premature convergence or slow search progress. To address these challenges, this paper proposes an adaptive dual-module competition and cooperation (ADCC) framework. The framework enables microscopic cooperation and macroscopic competition between the exploration and exploitation modules, thereby achieving more flexible adjustment between the two search behaviors. The two modules first perform adaptive fusion at the dimension level based on historical performance to generate a fused solution. Subsequently, under the macroscopic competition mechanism, the current search requirements of the population are comprehensively assessed, and an adaptive selection is made between the individual module solutions and the fused solution. Through comprehensive performance comparisons — including ADCC for low-level strategy fusion, ADCC for high-level algorithm integration, and ADCC-based DE (ADCCDE) against state-of-the-art DE variants, we rigorously validate and analyze the effectiveness and advantages of the proposed framework on both benchmark functions and real-world problems.

1. Introduction

Since its proposal by Storn and Price (Storn and Price, 1997), differential evolution (DE) algorithm has rapidly become one of the most popular optimization techniques in the field of evolutionary computation (EC) due to its simplicity, ease of implementation, and strong global search ability (Price et al., 2005). Its core idea stems from swarm intelligence, simulating the biological evolution process through mutation, crossover, and selection operators to drive the population towards the optimal solution. DE has been successfully applied in numerous complex engineering and scientific fields, including economic dispatch in power systems (Biswas et al., 2019a), image processing (Tang et al., 2024), fuel cell parameter estimation (Jangir et al., 2025), medical diagnosis (Kaur et al., 2019; Srivastava and Pradhan, 2023), and planning of renewable energy systems (Biswas et al., 2019b; Abd El-Mageed et al., 2023), demonstrating outstanding robustness and wide applicability in problem-solving.

With the continuous increase in the complexity of optimization problems in modern application fields (such as high dimensionality, multi-modality, and nonlinearity), the search performance of the DE algorithm faces higher demands. How to dynamically balance exploration and exploitation capabilities during the algorithm's iteration process and how to efficiently and adaptively regulate strategies and parameters remain the core challenges in enhancing DE performance (Reyes-Davila et al., 2025). Exploration aims to expand the search range to discover potential optimal regions, while exploitation focuses on refining the search in known high-quality regions. An imbalance between the two can lead the algorithm to get stuck in local optima or converge slowly. To address this challenge, researchers have attempted to improve DE performance by designing specific mechanisms or schemes. Notably, a scheme often excels only in specific problems or at certain stages of the iteration due to its inherent design bias towards exploration or exploitation. As the "No Free Lunch Theorem" (Wolpert and Macready, 1997) reveals, no single scheme can satisfactorily solve all problems. Exploration-biased schemes may be efficient in the early stages of complex multi-modal problems but tend to lose efficiency due to excessive dispersion in the later convergence stage; while exploitation-biased schemes, although accelerating local

* Corresponding author.
E-mail address: zhangsx@jnu.edu.cn (S.X. Zhang).
[1] Y.X. Wu and Y.H. Wang contributed equally to this work.

https://doi.org/10.1016/j.engappai.2026.115602
Received 31 August 2025; Received in revised form 28 January 2026; Accepted 1 July 2026
Available online 13 July 2026

optimization, often struggle to meet the global search requirements in high-dimensional problems. For this reason, the switching or combination of different schemes with different parameter settings can better solve specific problems than a single scheme. This has led some researchers to focus on the integration of new methods or the cooperation of different schemes to achieve fine-grained control over the balance between exploration and exploitation, thereby enhancing the performance of DE.

Existing methods include generating candidate solutions from a predefined strategy pool (Yi et al., 2022) based on the fitness landscape and selecting the optimal offspring. There are also approaches that dynamically adjust the usage probability of complementary operators (Sun et al., 2020), local and global parameters (Jin et al., 2025) and subpopulation size (Nadimi-Shahraki and Zamani, 2022) according to historical success rates, so as to adaptively meet the requirements of exploration and exploitation in different optimization stages. Additionally, some researchers have employed the concept of reinforcement learning to optimize the parameter adjustment process (Tatsis and Parsopoulos, 2023), enabling the algorithm to adapt to the optimal parameter settings for different optimization problems. Such methods can be categorized as "competitive-based DE". The corresponding alternative category is "cooperative-based DE". These methods integrate different schemes according to population characteristics (Huang et al., 2025) and evolutionary stages (Liao et al., 2024), and collaboratively complete the entire iterative process.

In current research, whether through coordinating multiple strategies in different scenarios or comprehensively considering the characteristics of different schemes to select the optimal one, the final offspring is generated from the original single strategy. They mostly remain at the individual selection level and rarely dynamically integrate the advantages of two modules to form a new integrated individual and incorporate it into the competitive system. This leads to a common problem in the exploration–exploitation transition stage of the algorithm—when the offspring of a single strategy has difficulty balancing the demands of global search and local convergence, there is a lack of transitional solutions that fuse the advantages of both. Information exchange among candidate offspring can integrate the advantages of different strategies or mechanisms into a single offspring, performing gene fragment cutting and recombination at the individual level to form potentially more suitable solutions, which is conducive to solving problems with different natures during the iterative process and better connecting the exploration and convergence stages. Motivated by these observations, this paper proposes an adaptive dual-module competition and cooperation (ADCC) framework. The proposed framework is distinguished by the following key features:

(1) By designing two basic modules with complementary characteristics and their fusion scheme, the adaptive regulation of exploration and exploitation capabilities is achieved. One module tends to global exploration and expands the search space through diversified strategies or candidate solutions with large Euclidean distance from the parent, while the other focuses on local exploitation and accelerates convergence by utilizing elite information or candidate solutions with small Euclidean distance from the parent.

(2) The two modules dynamically integrate the core operations of exploration and exploitation based on the fitness of the parent and the current iteration progress, achieving the cutting of gene fragments at the dimensional level.

(3) To achieve the on-demand selection of the three, a dynamic threshold is introduced. This threshold comprehensively considers the population diversity index and the iteration progress, tending to select the exploratory module in the early stage of evolution to maintain diversity, balancing exploration and exploitation through fusion scheme in the middle stage, and switching to the exploitable module in the later stage to accelerate convergence.

The rest of this paper is organized as follows. Section 2 briefly introduces the classic DE algorithm and reviews the related works. Section 3 describes the proposed ADCC optimization framework in detail. Section 4 presents comprehensive experiments to validate the proposed method while Section 5 concludes this paper.

2. Background

2.1. Differential evolution

DE (Storn and Price, 1997) is a population-based stochastic search algorithm which utilizes the differential information of the population to approach the optimum. The working steps of DE are as follows.

Initialization: DE begins with a randomly initialized population consisting of NP individuals $\{\mathbf{x}_{i,0} = (x_{i,1,0}, x_{i,2,0}, \ldots, x_{i,D,0}), i \in 1, 2, \ldots, NP\}$, each individuals is constrained within $[x_j^{low}, x_j^{high}]$, where $j \in 1, 2, \ldots, D$. It then performs a loop of genetic operators until the stopping criteria are met.

Mutation: Following initialization, mutation is conducted on the target population $\mathbf{x}_{i,g}$ to generate a perturbed population $\mathbf{v}_{i,g}$ at generation g. The classic "DE/rand/1" mutation is described in Eq. (1).

$$\mathbf{v}_{i,g} = \mathbf{x}_{r1,g} + F \cdot (\mathbf{x}_{r2,g} - \mathbf{x}_{r3,g}) \tag{1}$$

where $\mathbf{x}_{r1,g}$, $\mathbf{x}_{r2,g}$ and $\mathbf{x}_{r3,g}$ are three individuals randomly selected from the current population; $r1, r2$ and $r3$ are random integers within $[1, NP]$ with $r1 \neq r2 \neq r3 \neq i$. Mutation factor F is a user-defined control parameter within $(0, 1]$.

In Zhang and Sanderson (2009), the strategy "DE/current-to-pbest/1" for learning from local elites is proposed. This ensures diversity of strategies, balances the algorithm between exploitation and exploration, and reduces greed. The mutation operator is described in Eq. (2).

$$\mathbf{v}_{i,g} = \mathbf{x}_{i,g} + F \cdot (\mathbf{x}^p_{best,g} - \mathbf{x}_{i,g}) + F \cdot (\mathbf{x}_{r1,g} - \mathbf{x}_{r2,g}) \tag{2}$$

where $\mathbf{x}^p_{best,g}$ is the top $p\%$ of elite individuals in the current population. For the current individual, a random elite is selected as part of the difference to guide the individual's direction of progress.

The learning strategy of this elite collective has been extended in Zheng et al. (2017), integrating collective information from m best candidates to form a new strategy "DE/current-to-ci_$m$best/1" operator, specifically in Eq. (3).

$$\mathbf{v}_{i,g} = \mathbf{x}_{i,g} + F \cdot (\mathbf{x}_{ci\_mbest,g} - \mathbf{x}_{i,g}) + F \cdot (\mathbf{x}_{r1,g} - \mathbf{x}_{r2,g}) \tag{3}$$

Where $\mathbf{x}_{ci\_mbest,g}$ is a candidate information obtained through linear combination.

Crossover: Once the perturbed population $\mathbf{v}_{i,g}$ is generated, the crossover operator is employed to generate a trial population $\mathbf{u}_{i,g}$. The crossover operator is defined as:

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

where j_{rand} is a randomly generated integer within $[1, D]$ and Cr is the crossover factor within $[0, 1]$. Through crossover, the mutant vector $\mathbf{v}_i$ and the target vector $\mathbf{x}_i$ undergo dimensional fusion to form the trial vector $\mathbf{u}_i$ with genetic diversity. The constraint $j = j_{rand}$ ensures that at least one dimension in the individual is derived from $\mathbf{v}_i$.

Selection: After crossover, $\mathbf{u}_i$ is evaluated and compared with the corresponding $\mathbf{x}_i$. The one with smaller fitness is then selected for the next generation (for minimization problems) as follow:

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

2.2. Cooperative DE

Over the past two decades of research, DE has attracted extensive attention due to its simple three-step core framework and efficient computational advantages, and has derived various extensions; the combinations of multi-strategy, multi-population, and so on have been noted, and the features of the classification of the aforementioned problems, along with the proposed method in this study, are collated in Table 1. These can be collectively termed the cooperative DE,

Table 1
A summary of Cooperative DE, Competitive DE and ADCCDE

Mechanism	Type	Algorithm	Feature	Year
Cooperative	Multi-stage	IDE (Tang et al., 2014)	Stage-specific parameters and mutation operators	2014
		EaDE (Zhang et al., 2021)	Selective-candidate with Similarity Selection and adaptive stages	2021
		MIDE (Gupta et al., 2023)	Stage-wise strategy reset and multi-elite guided mutation	2023
		DACDE (Chen et al., 2024)	Stage-varying scaling factor and differentiated mutation	2024
		MSDE (Cai et al., 2024)	Stagnation-based three-stage strategy selection	2024
	Multi-population	PaDE (Meng et al., 2019)	Each group maintains independent param/prob	2019
		MPMSDE (X. Li et al., 2021)	Biased Resource-Allocation Strategy-Ranked Subpopulations	2021
		DDE (J.-Y. Li et al., 2022)	Parallel evolution with dedicated strategy per subpopulation	2022
		APPDE (Wang and Ma, 2023)	"Main population + Auxiliary population" structure	2023
		DyS-MPADE (Huang et al., 2025)	Clustering-based dynamic subpopulation number adjustment	2025
Competitive	Successful experiences	SaDE (Qin et al., 2008)	Strategy candidate pool with success-history-based selection probabilities	2008
		MVC (Zhang et al., 2017)	Resource bias toward high-success-rate subpopulations	2017
		CJADE (Gao et al., 2019)	Heterogeneous chaotic local search with memory-based dynamic selection	2019
		DMDE (Nadimi-Shahraki and Zamani, 2022)	Adaptive subpopulation size adjustment via success rates	2022
		SRADE (Qiao et al., 2022)	Competitive resource allocation via cumulative strategy success rates	2022
		EA4eig (Bujok and Kolenovsky, 2022)	Roulette-wheel competition for execution rights via solution-based success rates	2022
		ESADE (Zhang et al., 2024)	Step-size estimation via successful vector distance for scale indicator updates	2024
	Fitness landscape	FDCDE (W. Li et al., 2021)	Fitness-quantified landscape characteristics for dynamic strategy selection	2021
		EJADE (Yi et al., 2022)	Dual trial-vector mechanism with success-based competitive selection of operators/parameters	2022
		MMFLDE (Li et al., 2023)	Fitness landscape characteristics-driven optimal strategy matching for current search stage	2023
		LSHADE-Code (Chen et al., 2025)	Random parameter assignment per individual with best-fitness trial vector survival	2025
		FLADE (Tang et al., 2026)	Success-history-based competitive activation of lofting-guided strategy for directed search	2026
	Reinforcementlearning	RL-HPSDE (Tan et al., 2022)	Individual state observation, param/strategy selection, effectiveness-based rewards	2022
		RL-DAS (Guo et al., 2024)	Landscape/history-driven feedback, cost/convergence rewards, inter-algorithm dynamic competition	2024
		RLDEA (Yu et al., 2025)	Feedback selection via offspring-parent fitness ratio	2025
		AuDE (Cao et al., 2025)	Reward mechanism via promotion/success rates for experience-based autonomous strategy selection	2025
		IRLDE (Wang et al., 2025)	Fitness-feedback dynamic selection, Gaussian/roulette guidance	2025
Cooperative and competitive	——	ADCCDE	Microscopic dimension fusion and macroscopic population competition	——

summarized into the following classifications:

2.2.1. Multi-stage-based

Most multi-stage approaches partition different stages or determine different stage states to match the optimal strategy for the current stage. In Tang et al. (2014), based on an individual's own evolutionary state at different stages, suitable mutation strategies are matched. In Zhang et al. (2021), a Selective-candidate with Similarity Selection (SCSS) stage and an adaptation scheme stage are proposed in a collaborative manner. In the subsequent stage, strategies are adaptively selected based on the learning outcomes from the SCSS stage. In Gupta et al. (2023), the iterative process is divided into stages according to a preset generational interval, with parameter reset and strategy adjustment triggered at each stage. In Chen et al. (2024), the optimization process is partitioned into early and late stages, different dynamic adaptation is applied to parameters per stage, enhancing exploration capability at the early stage and strengthening local exploitation at the late stage. In Cai et al. (2024), a dynamic multi-stage selection mechanism is designed, and stagnation counts are divided by stage, activating different selection strategies sequentially.

2.2.2. Multi-population-based

By designing cooperative methods within a single population or parallel multi-populations, multi-populations can enhance algorithm diversity. Meng et al. (2019) divided the population into groups, with each group independently maintaining control parameters and selection probabilities. X. Li et al. (2021) proposed that subpopulation sizes be dynamically allocated according to strategy performance rankings, with higher-ranked strategies obtaining larger subpopulation sizes. Wang and Ma (2023) developed a "main population + auxiliary population" dual-population structure, where the auxiliary population serves to archive suboptimal solutions, while both populations generate trial vectors and participate in offspring selection. J.-Y. Li et al. (2022) employed multiple populations for parallel evolution, with different strategies deployed to each population. Limited evaluation resources are allocated to better-performing populations based on performance metrics. Huang et al. (2025) adopted an explicit multi-population architecture, analyzing population distribution via clustering to divide subpopulations and promote cooperation between them.

2.3. Competitive DE

Competitive DE which is implemented through contention for limited computational resources, has consistently attracted significant research attention. These paradigms are presented in Table 1 for comparison with our proposed method, with detailed classifications elaborated as follows:

2.3.1. Successful experiences-based

For algorithms to converge toward the global optimum, the guidance from successful individuals is essential. In Qin et al. (2008), a competitive strategy selection based on historical success probabilities is developed. Heterogeneous chaotic local search operators are integrated in Gao et al. (2019), with selection probabilities dynamically adjusted via a memory mechanism that computes the success rate of each chaotic mapping. In Zhang et al. (2017), candidate optimizers run independently in each segment and the optimal one is then selected based on the cumulative optimization efficiency of fitness improvement. The size of subpopulations is determined by their respective success rates during iteration in Nadimi-Shahraki and Zamani (2022), with larger sizes and more computational resources allocated to those with higher success rates. In Qiao et al. (2022), a method where computational resources are dynamically allocated to each strategy based on its cumulative success rate over past iterations is introduced. In Bujok and Kolenovsky (2022), a multi-algorithm cooperative framework, where selection probabilities of algorithms are dynamically adapted via success-count-based roulette wheel selection is constructed. In Zhang et al. (2024), an innovative metric which is based on the evolutionary scale of successful individuals is proposed with strategies selected via this metric.

2.3.2. Fitness landscape-based

Direct observation of fitness landscape changes or computation of indicators based on fitness increments is an alternative approach for characterizing problem properties. W. Li et al. (2021) proposed measuring landscape characteristics by quantifying individual fitness, with appropriate strategies dynamically selected accordingly. Yi et al. (2022) applied a dual trial vector mechanism to each individual, where the fitness values of two trial vectors are compared and the superior one is preserved for the next generation. Li et al. (2023) utilized the features

of the fitness landscape to determine the currently suitable strategy for the subsequent construction of the operator selector. Chen et al. (2025) set three mutation strategies, with parameter combinations randomly assigned to each individual, and only the trial vector with the optimal fitness advancing to the next generation. Tang et al. (2026), through fitness landscape sequence analysis, unbiased kurtosis estimator for detecting local optimum entrapment, and lofting guidance strategy.

2.3.3. Reinforcement learning-based

The reinforcement learning can dynamically control and coordinate the running direction of the algorithm in real time in constant feedback. A model-free time-difference reinforcement learning algorithm is used in Tan et al. (2022), where each individual observes the environmental state, selects parameter and mutation strategy combinations, and receives rewards based on their effectiveness. In Guo et al. (2024), based on landscape characteristics and algorithm history extraction status, reinforcement learning feedback with cost reduction and convergence speed as rewards, a dynamic competition mechanism between algorithms is designed. The Q-learning reinforcement learning method is commonly used as the basis for competitive selection of parameters and policies. In Yu et al. (2025), the feedback selection is carried out according to the fitness ratio of the offspring and the parent. In Cao et al. (2025), a reward mechanism based on individual promotion rate and strategy success rate was designed, so that the algorithm can independently select strategies according to historical accumulated experience. In Wang et al. (2025), an improved Q-learning algorithm is proposed to perform dynamic competitive selection according to fitness feedback in each iteration, under the guidance of Gaussian distribution and roulette-wheel selection mechanism.

3. Proposed method

3.1. Motivation

As reviewed in Section 2, in the research of DE, effective strategies for balancing exploration and exploitation are mainly classified into two categories: cooperative and competitive approaches. The cooperative approaches, as shown in Fig. 1(a), focus on integrating multi-source information (such as subpopulation interaction in multi-population cooperation and strategy cooperation in multi-stage cooperation); Fig. 1(b) summarizes the competitive mechanism, which essentially achieves resource optimization through dynamic strategy selection (such as fitness landscape, successful experiences, and reinforcement learning). Despite the significant achievements of these cooperative and competitive approaches, they still have some common limitations:

(1) Although the existing competitive or cooperative mechanisms can dynamically select superior strategies, most methods generate candidate offspring independently from a single original strategy when balancing exploration and exploitation, without constructing a collaborative solution that integrates the advantages of multiple strategies and enables their participating in the competition. An idealized assumption for the population state is that a population is either in the state of exploration or exploitation. In reality, the population state undergoes continuous variation rather than phase-wise variation. When the population transitions from exploration to exploitation, it may be in a state where the global distribution has already exhibited obvious contraction, but the distance differences between individuals remain significant and a stable local structure has not yet formed. The existing state of adaptive generation of a single subpopulation and continuous evolution is difficult to match during the instantaneous replacement of subpopulations. When the algorithm is in the transition stage between exploration and exploitation, the original multiple strategies are not well adapted.

(2) The strategies or mechanisms corresponding to the exploration and convergence stages of most algorithms are usually preset and fixed. When the exploration and exploitation characteristics of the selected strategy are ambiguous, the preset is often inaccurate and lacks a real-time redistribution mechanism. The "static" mode makes it difficult for the algorithm to adapt to the dynamic changes during the optimization process.

To address these issues, this paper proposes an adaptive dual-module competition and cooperation (ADCC) optimization framework, whose core idea is "dimensional dynamic cooperation and competitive selection", as shown in Fig. 1(c): Firstly, an exploration module and an exploitation module are designed. They not only independently generate offspring but more importantly, they adaptively interact and integrate at the dimension level to produce a third type of offspring representing the cooperative effect. Then, a threshold based on the Lehmer formula is introduced for dynamic update. This threshold comprehensively considers historical search performance, the current evolutionary stage, and population status to adaptively select the optimal offspring from the three types of offspring. Ultimately, it achieves the dynamic switching and balance among three modes: exploration dominance, transitional integration, and exploitation dominance throughout the entire optimization process.

3.2. Collaborative framework

To address the problem of mismatch between the instantaneous nature of exploitation–exploration transition and the population state, the algorithm integrates the respective advantages of exploitation and exploration during the collaboration phase, thus generating a transition subgroup. Since ADCC can be embedded with strategies and algorithms, when operating in the algorithm framework, we encapsulate such algorithms or strategies in the form of modules to accommodate flexible input requirements. The exploitation candidate population generated by the exploitation module is denoted as **UT**, and the exploration candidate population generated by the exploration module is denoted as **UP**. The transition subgroup **UC** is formed by the micro-integration of these two modules.

For the input of modules, exploitation and exploration are generally automatically distinguished via preset configurations and then input into the exploitation module and exploration module respectively, but since it is sometimes difficult to clearly distinguish the characteristics of these two input modules, this algorithm adopts an adaptive method for such distinction. This adaptive method is based on Euclidean distance, and since exploration is intended to traverse unknown regions in the population and search for potential optimal solutions, exploratory individuals have dispersive characteristics, in contrast, exploitation focuses on local refinement to further improve solution quality, so exploitative individuals have aggregative characteristics.

Euclidean distance is used to calculate the spatial distances between individuals in the candidate trial populations **U1** and **U2** generated by the exploitation module and exploration module respectively and their corresponding parent individuals **x**, and we can determine whether an individual is dispersive or aggregative relative to its parent and whether it belongs to the exploitative or exploratory type by comparing these distances, with the specific distance calculation and judgment methods as follows:

$$d1_{i,g} = \|\mathbf{u1}_{i,g} - \mathbf{x}_{i,g}\|, d2_{i,g} = \|\mathbf{u2}_{i,g} - \mathbf{x}_{i,g}\| \tag{6}$$

The ith ($i \in 1, 2, \ldots, NP$) individual is classified according to the distance of $d1_{i,g}$, $d2_{i,g}$. Individuals with shorter distances, those relatively aggregated around their parent individuals, are incorporated into the exploitation candidate trial population, whereas individuals with longer distances from their parent individuals are incorporated into the exploration population:

$$\begin{cases} \mathbf{u1}_{i,g} \rightarrow \mathbf{ut}_{i,g}, \mathbf{u2}_{i,g} \rightarrow \mathbf{up}_{i,g}, & \text{if } d1_{i,g} < d2_{i,g} \\ \mathbf{u1}_{i,g} \rightarrow \mathbf{up}_{i,g}, \mathbf{u2}_{i,g} \rightarrow \mathbf{ut}_{i,g}, & \text{otherwise} \end{cases} \tag{7}$$

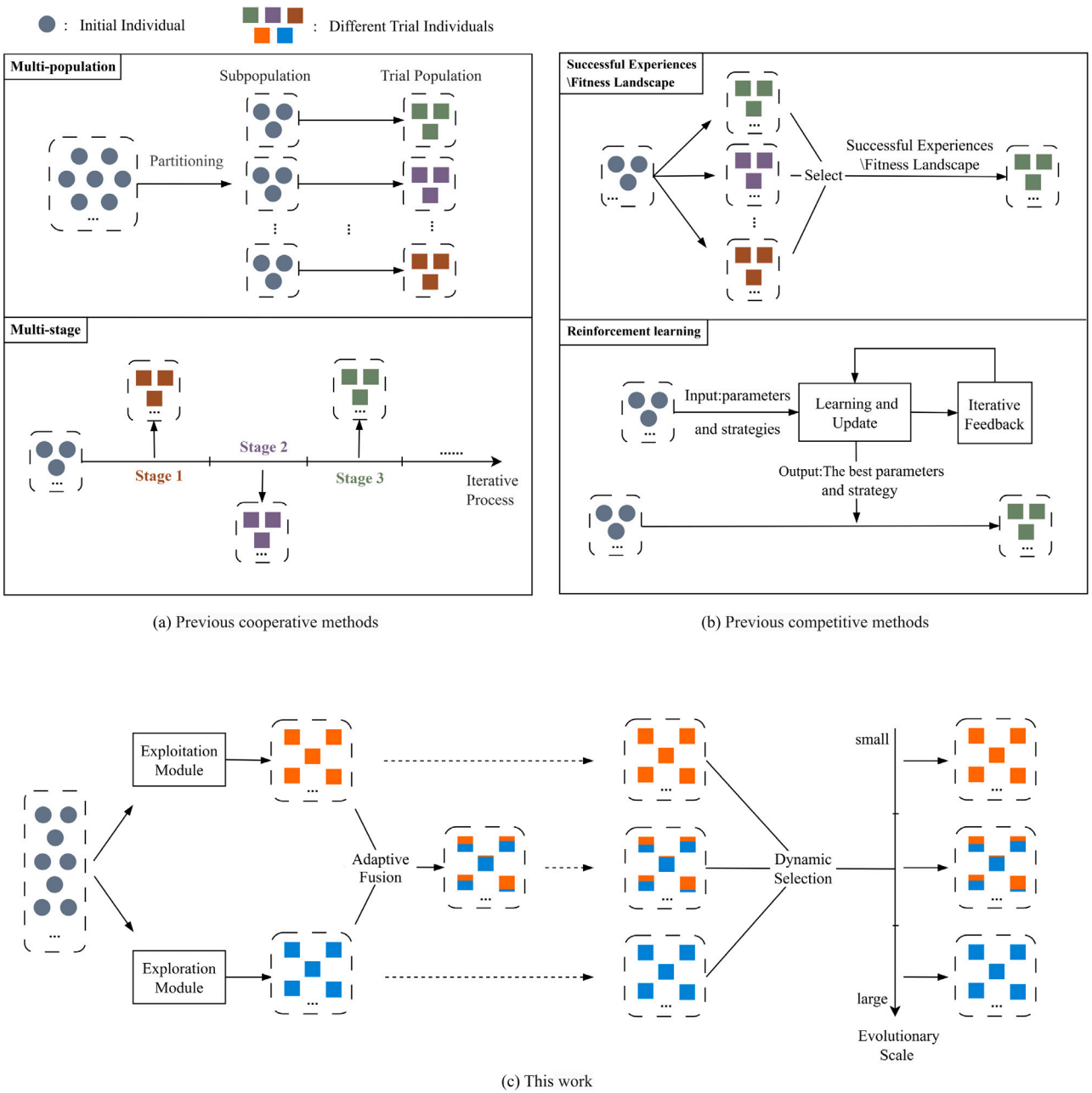


Fig. 1. Illustration of the proposed and the previous cooperation and competition methods.

The generation of **UC** is based on the exploitation and exploration candidate trial subgroups, we rank the dimensional diversity of the parent population at the individual dimensional level. The ranking value can represent the degree of aggregation and dispersion of the dimensions in the current parent population, which corresponds to the degree of exploitation and exploration. Then, we perform sorting and normalization on the fitness values of current individuals, and map the processed values to the individual dimensional level. Thus, each individual is assigned a dimensional threshold to denote its degree of requirement for exploitation and exploration, followed by the execution of dimensional fusion for exploitation and exploration. Initially, the diversity of each dimension in the parent population is calculated:

$$Div_{j,g} = \sqrt{\frac{1}{NP}\sum_{i=1}^{NP}(x_{i,j,g} - \overline{x}_{j,g})^2} \tag{8}$$

where $\overline{x}_{j,g}$ is the center of mass in the jth ($j \in 1, 2, \ldots, D$) dimension. The diversity of each dimension is calculated, sorted in ascending order, and the rankings are stored in R_{div} from 1 to D. The ranking value can reflect the relative degree of dispersion among dimensions. A higher ranking indicates greater diversity, which means a large degree of internal difference within the current dimension; conversely, it means a small degree of internal difference.

Fitness sorting is performed on the individuals of the current population to obtain the ranking denoted as $R_{fit}(i)$. The ranking value range of each individual is $[1, NP]$. The closer the value is to NP, the lower the ranking, and the larger the corresponding fitness value. Individuals with higher rankings are expected to continue exploiting their own advantages to guide the population towards an optimal direction. Meanwhile, individuals with lower rankings are also supposed to utilize their value to increase population diversity and further expand the exploration space.

Accordingly, we perform the fusion of **uc** at the individual dimensional level and allocate the dimensional proportion of individuals from the two candidate trial subgroups, the exploration subgroup **up**

and the exploitation subgroup **ut**, for each current individual. For individuals with smaller $R_{fit}(i)$ values, more dimensions from the convergent candidate trial population individuals will be allocated; for individuals with larger $R_{fit}(i)$ values, more dimensions from the exploitative candidate trial population individuals will be allocated. For the ith individuals of generation g of the fusion population **UC**, each dimension is selected according to the threshold $Th_{div}^{i,g}$, the calculation is as follows:

$$Th_{div}^{i,g} = floor((\frac{NP - R_{fit}(i) + 1}{NP}) \cdot D^{k}) \tag{9}$$

The $floor$ indicates that the computed threshold is rounded down. k is dynamic factor. Considering that the population needs to explore extensively through rich diversity in the early stage of evolution, and needs to focus on the deep development of high-quality local solution in the later stage, we further incorporate the influence of the iterative process and set it as the threshold of dynamic change. This dynamic factor is defined as follows:

$$k = 1 + (\frac{nfes}{MAX\_nfes}) \tag{10}$$

where $nfes$ is the number of current fitness evaluations, and MAX_nfes is the total number of evaluations. The threshold Th_{div} is derived by normalizing R_{fit} and mapping the normalized result to the dimensional level, which divides the exploitation and exploration of individuals at the dimensional level. The threshold is the outcome of inverse normalized mapping of R_{fit}, indicating the number of dimensions required for individual convergence. A low diversity ranking corresponds to a low degree of dispersion, meaning that the dimension is in a relatively convergent state and thus suitable for dimensional exploitation, and the reverse is also true. By comparing the diversity ranking of each dimension in the parent subgroup with the threshold of the current individual, we determine which dimensions are assigned to exploitation and which to exploration. Specifically, for the ith individual, when $R_{div}^{j,g} \leq Th_{div}^{i,g}$, all dimensional components in $\mathbf{x_{i,g}}$ that correspond to the dimensions with values below the convergence threshold will adopt the corresponding dimensional components of individuals from the exploitation module. Conversely, when $R_{div}^{j,g} > Th_{div}^{i,g}$, the corresponding dimensional components of individuals from the exploration module will be adopted.

As described in Fig. 2, for the individuals fused by the exploration and exploitation candidate trial populations, with the increase in individual fitness ranking, the proportion of dimension fragments from the exploration population will be more. The color increasingly appears as a deep shade of blue.

The aforementioned process is summarized in Algorithm 1. Line 1 specifies that two distinguishable modules generate the candidate experimental populations **UT** and **UP** via operations including mutation and crossover. If the modules are indistinguishable, Lines 2–3 execute the adaptive allocation of offspring for exploitation and exploration by means of Euclidean distance. Lines 4–7 show the calculation of population dimension diversity and its ranking. Line 8 calculates the dynamic factor k based on the iterative process, and forms the dynamic dimension selection threshold Th_{div} in line 10. In lines 11–17, for the ith individual, each dimensional components will select the fusion gene by comparing the current dimension threshold with the dimension diversity ranking. Ultimately, the fused population **UC** is obtained through microscopic collaboration.

Algorithm 1 Collaborative framework

Require: current population **X**; exploitation and exploratory module; population size NP; problem dimension D.
Ensure: Candidate trial populations **UT** and **UP**; integrated candidate trial population **UC**.

```
1: Candidate trial populations UT and UP are generated from the input exploitation and exploratory modules.
2: If the two modules are unknown, the Euclidean distances between their respective candidate trial populations U1, U2 and X are calculated using Eq. (6), yielding the distances D1 and D2.
3: D1 and D2 are allocated via Eq. (7) to obtain the exploitation and exploration candidate trial populations UT and UP.
4: for j = 1 : D do
5:     Calculate the dimensional diversity Div_{j,g} by Eq. (8);
6: end for
7: Sort Div_{j,g} in ascending order to obtain the diversity ranking R_{div};
8: Calculate the dynamic factor k by Eq. (10);
9: for i = 1 : NP do
10:    Calculate individual dimension selection threshold Th_{div}^{i,g} by Eq. (9);
11:    for j = 1 : D do
12:        if R_{div}^{j,g} ≤ Th_{div}^{i,g} then
13:            ut_{i,j,g} → uc_{i,j,g};
14:        else
15:            up_{i,j,g} → uc_{i,j,g};
16:        end if
17:    end for
18: end for
```

3.3. Competitive framework

The reasonable selection of trial vectors determines whether superior solutions can be found efficiently. In the competition, to avoid the candidate trial population being too random or too greedy, we adopt an adaptation mechanism based on the deviation degree of the successful individual's parent and offspring to evaluate the evolution scale, and the main index is the evolutionary scale factor. This strategy is referred to as the evolutionary scale adaptation (ESA) mechanism. The evolutionary scale factor (Zhang et al., 2024) is calculated as follows:

(1) At the beginning, calculate the Euclidean distance between the parent target vector $\mathbf{x}$ and the trial vector $\mathbf{u}$, sort them in ascending order from small to large, normalize them to the interval $(0, 1)$, and finally obtain the normalized distance rank R_{dist}, where the ranking in R_{dist} is sorted in ascending order.
(2) Then, for the ith individual, if the fitness of its trial vector is better than that of the target vector, $R_{dist}(i)$ is saved into the set λ.
(3) Next, σ reflects the evolutionary scale of the population at different stages, which is obtained by weighting the normalized distance ranking of successful individuals by Lehmer mean as follows:

$$\sigma = \frac{\sum_{m=1}^{|\lambda|} w_m \cdot \lambda_m^2}{\sum_{m=1}^{|\lambda|} w_m \cdot \lambda_m} \tag{11}$$

$$w_m = \frac{\Delta f_m}{\sum_{m=1}^{|\lambda|} \Delta f_m} \tag{12}$$

where Δf_m is the absolute value of the difference between the fitness value of the parent and the offspring of the mth individual.

On a small scale, the parent–child distance of the set of successful individuals is closer, which reflects that the whole population is moving towards the convergence trend. On the contrary, a large scale means that the parent–child generation of the successful individual set is far away. The evolutionary scale indicator γ is calculated using dynamic learning, where the learning factor c is set to 0.9:

$$\gamma = (1 - c) \cdot \gamma + c \cdot \sigma \tag{13}$$

Through extensive experimental evaluation, it is observed that the algorithm exhibits relatively poor overall performance when the learning

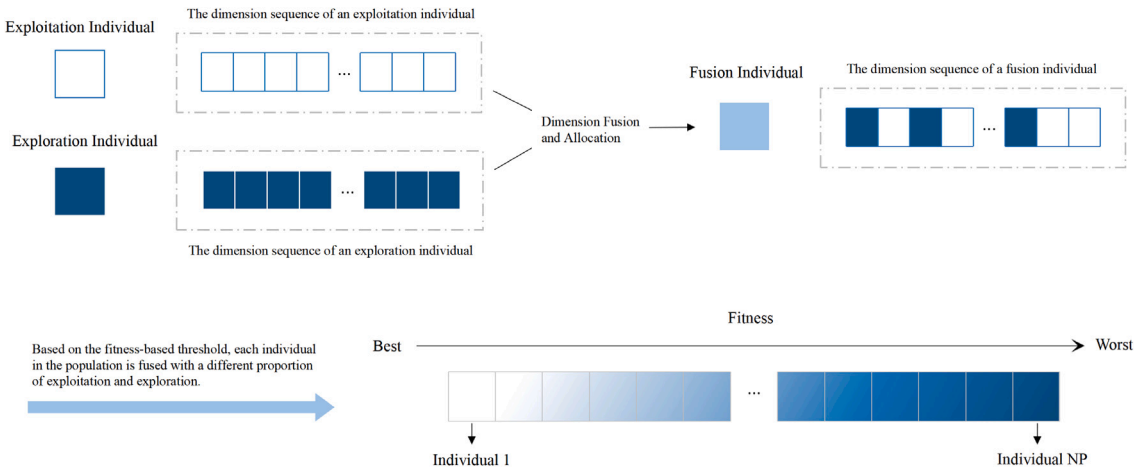


Fig. 2. Individual integration in exploration and exploitation, and individual integration variation with fitness.

factor c lies in the range of 0.1–0.6. In contrast, when c is set between 0.6 and 0.9, the algorithm achieves noticeably improved performance, which remains comparatively stable across this interval. The dynamic population macroscopic selection mechanism based on γ is used to select the trial population **U**. When γ is large, it indicates that in recent iterations, the successful trial vectors are distributed globally.

In this case, enhancing the exploration capability is more conducive to the current evolutionary state, so the exploratory candidate population **UP** is selected. Conversely, it suggests that the successful solutions in the current evolutionary process tend to be locally distributed, and there is a need to strengthen the local exploitation capability, thus selecting the exploitation candidate population **UT**. For the transition region between the above two stages, the fusion population **UC** that fuses the advantages of the first two stages is used. The interval threshold increases linearly with the iteration process, enabling extensive exploration in the early stage and fine exploitation in the later stage, thereby guiding the algorithm toward the global optimum and accommodating different optimization needs:

$$U = \begin{cases} UP, & \text{if } d_{high} \le \gamma \\ UC, & \text{if } d_{low} \le \gamma < d_{high} \\ UT, & \text{otherwise} \end{cases} \tag{14}$$

$$d_{low} = d_{th}^{L} + 0.1 * (nfes/MAX\_nfes) \tag{15}$$

$$d_{high} = d_{th}^{H} + 0.1 * (nfes/MAX\_nfes) \tag{16}$$

Where d_{th}^{L} is set to 0.45 and d_{th}^{H} is set to 0.55. As the iterations progress toward the final generation, the dynamic threshold gradually converges to the range $[0.55, 0.65]$, thereby increasing the selection probability of the exploitation candidate trial population and accelerating convergence in the later stage. The initial threshold has also undergone extensive experimental tests, and both excessively small and excessively large initial thresholds yield poor results in performance tests. When the initial threshold is excessively small, the exploratory population **UP** will only be selected when the evolutionary scale is extremely small, which makes it difficult for the algorithm to escape when it gradually falls into local optimum. Conversely, an excessively large initial threshold will hinder the convergence performance. The competition process is described in lines 2–9 in Algorithm 2.

3.4. Overall implementation

The overall framework of the adaptive dual-module competition and cooperation-based DE (ADCCDE) is shown in Algorithm 3. Line 1 initializes the population, and after entering the internal iteration cycle. In line 3, the candidate trial populations $\mathbf{UT}_g$, $\mathbf{UP}_g$ are first acquired,

Algorithm 2 Competitive framework

Require: population size NP; evolutionary scale indicator γ.
Ensure: trial population **U**; evolutionary scale indicator γ.
1: Obtain candidate trial populations **UT**, **UP** and integrated candidate population **UC** by **Algorithm** 1;
2: Calculate the evolutionary scale threshold d_{low} and d_{high} by Eqs. (15) and (16);
3: **if** $d_{high} \le \gamma$ **then**
4: **UP** → **U**;
5: **else if** $d_{low} \le \gamma < d_{high}$ **then**
6: **UC** → **U**;
7: **else**
8: **UT** → **U**;
9: **end if**
10: Calculate the Euclidean distance between **X** and **U**, sort distances in ascending order, and normalize them to get the distance ranking R_{dist};
11: **for** $i = 1 : NP$ **do**
12: **if** $f(u_{i,g}) \le f(x_{i,g})$ **then**
13: $R_{dist}(i) \rightarrow \lambda$;
14: **end if**
15: **end for**
16: Calculate evolutionary scale factor σ by Eq. (11);
17: Update evolutionary scale indicator γ by Eq. (13).

after which an integrated candidate population $\mathbf{UC}_g$ is generated by integrating the two sub-populations. In line 4, Algorithm 2 generates the trial population **U** and updates the evolutionary scale factor γ.

3.5. Novelty of ADCC

This paper presents an ADCC optimization framework, breaking through the limitations of existing collaborative/ competitive methods. It constructs a unified collaborative–competitive optimization paradigm composed of three key components: scalable modular architecture, dynamic role adaptation, and dimension reorganization, providing a more robust adaptive solution for complex optimization problems and better transitioning between exploration and exploitation phases. Specifically, we would like to highlight the novelty and contribution of the proposed ADCC method.

(1) Breaking away from the conventional paradigm in which a single module independently generates offspring, a dimension recombination–based fusion mechanism is proposed. The fusion

Algorithm 3 ADCCDE

Require: exploitation and exploratory module; set the initial evolutionary scale indicator $\gamma = 0.5$; maximum number of generations $Gmax$.

Ensure: found best solution $\mathbf{x}_b$.

1: Initialize the population $\mathbf{X}_0$, set the generation count $g = 0$;
2: **while** $g \leq Gmax$ **do**
3: Perform **Algorithm** 1 to obtain candidate trial populations $\mathbf{UT}_g$, $\mathbf{UP}_g$ and the integrated candidate population $\mathbf{UC}_g$;
4: Perform **Algorithm** 2 to obtain the trial population $\mathbf{U}_g$ and update evolutionary scale indicator γ;
5: Perform DE's selection between $\mathbf{X}_g$ and $\mathbf{U}_g$ to obtain the population $\mathbf{X}_{g+1}$ for the next generation;
6: $g = g + 1$;
7: **end while**
8: Obtain the best solution $\mathbf{x}_b$ from the final population.

ratio between two core modules is adaptively adjusted according to both the fitness information of the parent population and the iteration progress, thereby exploiting the complementary strengths of different strategies. By further introducing a fusion module, a three-party competitive structure is formed, establishing a novel "independent–fusion" collaborative–competitive paradigm.

(2) An adaptive role assignment mechanism is developed for the dual modules. When a module exhibits clear prior separability, it is explicitly designated as either exploration-oriented or exploitation-oriented. In contrast, when the functional boundary of a module is ambiguous or evolves over iterations, its role is dynamically determined using a Euclidean distance–based metric. This design ensures that each module is consistently aligned with the current search requirements.

(3) Beyond the use of fixed thresholds, an iteration-dependent dynamic variable is introduced to more accurately characterize the stage-wise behavior of the optimization process. This mechanism enables the algorithm to adapt to the exploratory nature in the early stages and the exploitative nature in the later stages of evolution, thereby improving the precision and effectiveness of the exploration–exploitation balance.

(4) Compared with existing cooperative or competitive optimization schemes, ADCC introduces a more flexible collaborative mechanism. Rather than relying on reinforcement-learning–driven or learning-based decision processes that induce instantaneous strategy switching, ADCC promotes smooth behavioral transitions that are more consistent with the current population state. In particular, collaborative subgroups are formed via dimension-wise feature assignment and fusion, allowing the algorithm to effectively accommodate intermediate transition phases where neither pure exploration nor pure exploitation is optimal.

4. Simulation

To verify the effectiveness of the proposed ADCC framework, relevant experiments are conducted in this section. The CEC2017 benchmark suite (Awad et al., Nanyang Technological University, Singapore, Tech. Rep. 2016) containing 29 functions is employed for the experiment. Among them, F1 and F3 are unimodal functions, F4–F10 are simple multimodal functions, F11–F20 are hybrid functions, and F21–F30 are composition functions. For each algorithm and each test function, 51 independent runs are conducted under the dimensions of 30, 50, and 100, respectively. A total of $10{,}000 \times D$ function evaluations are set as the termination criterion. The function error value $f(\mathbf{x}) - f(\mathbf{x}^*)$ is recorded to measure the performance of each algorithm, where x represents the best solution found during the operation of the algorithm, and x* denotes the global optimal solution of the test function.

To draw statistically sound conclusions when comparing two algorithms, 51 trials are conducted for each function, and Wilcoxon rank-sum test (Sheskin, 2003) is used to test the 51 items with a significance level of 0.05. The symbols "+", "≈" and "−" respectively indicate that the ADCCDE algorithm is significantly superior (i.e., win, W), comparable (i.e., tie, T), or worse than (i.e., lose, L) the compared algorithms.

4.1. ADCC for low-level strategy fusion

To verify the effectiveness of the ADCC framework in a single algorithm, two competitive baseline algorithms with large population, jSO (Brest et al., 2017) and LSHADE-cnEpSin Awad et al. (2017), are selected as verification carriers. Since large population algorithms require more efficient convergence mechanisms, we specifically choose two relatively convergent mutation strategies, "current-to-pbest" (Zhang and Sanderson, 2009) and "current-to-ci_*m*best" (Zheng et al., 2017). The former guides the population to explore new regions through the current superior solutions, undertaking the exploration module function; the latter, based on the weighted update of individual fitness rankings, strengthens local convergence and belongs to the exploitation module. According to the dual-module cooperation and competition scheme described in Section 3, these two strategies are respectively integrated into the baseline algorithms to construct two variant algorithms, ADCC-jSO and ADCC-LSHADE-cnEpSin.

(1) From Table 2, ADCC-jSO outperforms jSO across 30-D, 50-D, and 100-D cases, with winning counts of 13, 12, and 21, and losing counts of 3, 3, and 0 respectively. For unimodal and simple multimodal functions, ADCC-jSO achieves superior performance on five functions in both 30-D and 50-D. Specifically, these functions include 30-D F5–F8 and F10, 50-D F4, F5, F7, F8, and F10. When the dimension increases to 100, ADCC-jSO outperforms jSO on all functions except 100-D F1 (where their performance is comparable). Among hybrid functions, ADCC-jSO only lags behind jSO on 50-D F18; it performs better than jSO on 4 functions at both 30-D and 50-D. With the increase in dimension, this advantage further expands, and ADCC-jSO outperforms jSO on 6 functions on 100-D. For composition functions, the "W/L" of ADCC-jSO versus jSO are "4/2" and "3/1" on 30-D and 50-D, respectively. When the dimensionality increases to 100-D, the performance gap between the two widens, and the "W/L" shifts to "7/0".

(2) From Table 4, ADCC-LSHADE-cnEpSin wins in 11, 15 and 14 cases on 30-D, 50-D and 100-D functions respectively, and loses in 5 cases (including 30-D F4 and F25, 50-D F4, F25 and F28) out of the total 87 cases. Among unimodal and simple multimodal functions, ADCC-LSHADE-cnEpSin exhibits better performance than LSHADE-cnEpSin on 30-D, 50-D and 100-D F5–F8; in addition, its performance on 50-D and 100-D F10 is also improved. For hybrid functions, the performance of ADCC-LSHADE-cnEpSin is either superior or comparable to that of LSHADE-cnEpSin, with "W/L" of "3/0" on both 30-D and 100-D, and "5/0" at 50-D. Regarding composition functions, ADCC-LSHADE-cnEpSin shows slightly better performance than LSHADE-cnEpSin at 30-D and 50-D, with corresponding "W/L" of "4/1" and "5/2". When the dimensionality increases to 100-D, its performance is further enhanced, achieving a "W/L" of "6/0".

From the above comparisons with the two baseline algorithms, it can be seen that ADCCDE exhibits excellent performance on complex problems such as hybrid functions and composition functions. Due to the weighted summation of multiple functions, hybrid functions require grouped optimization of sub-components according to their characteristics (Awad et al., Nanyang Technological University, Singapore, Tech. Rep. 2016). In contrast, composition functions need to take into account the weighted fusion of basic functions. This requires the algorithm to possess a certain balanced coordination ability between

Table 2
Performance comparisons of ADCC–jSO with jSO on 30–D, 50–D and 100–D functions.

Problem	30–D					50–D					100–D				
	jSO			ADCC-jSO		jSO			ADCC-jSO		jSO			ADCC-jSO	
	mean	std	sig	mean	std	mean	std	sig	mean	std	mean	std	sig	mean	std
F1	0.00e+00	0.00e+00	≈	0.00e+00	0.00e+00	0.00e+00	0.00e+00	≈	0.00e+00	0.00e+00	0.00e+00	0.00e+00	≈	0.00e+00	0.00e+00
F3	0.00e+00	0.00e+00	≈	0.00e+00	0.00e+00	0.00e+00	0.00e+00	≈	0.00e+00	0.00e+00	3.49e−06	2.68e−06	+	3.62e−07	5.64e−07
F4	5.86e+01	0.00e+00	−	5.86e+01	3.00e−14	5.85e+01	5.04e+01	+	5.17e+01	4.34e+01	1.99e+02	1.52e+01	+	2.01e+02	1.09e+01
F5	7.43e+00	2.29e+00	+	5.60e+00	2.27e+00	1.36e+01	3.31e+00	+	1.08e+01	2.50e+00	3.67e+01	8.37e+00	+	2.73e+01	8.02e+00
F6	1.07e−08	3.72e−08	+	6.43e−08	2.16e−07	2.29e−07	4.06e−07	−	1.04e−06	1.60e−06	7.69e−05	1.25e−04	+	1.14e−04	3.69e−04
F7	3.83e+01	2.04e+00	+	3.65e+01	1.28e+00	6.67e+01	3.43e+00	+	6.14e+01	1.63e+00	1.44e+02	8.67e+00	+	1.30e+02	5.17e+00
F8	8.17e+00	1.98e+00	+	6.11e+00	2.32e+00	1.52e+01	2.92e+00	+	1.08e+01	2.89e+00	3.68e+01	8.43e+00	+	2.64e+01	7.69e+00
F9	0.00e+00	0.00e+00	≈	0.00e+00	0.00e+00	0.00e+00	0.00e+00	≈	0.00e+00	0.00e+00	1.94e−02	7.96e−02	+	5.68e−02	1.68e−01
F10	1.69e+03	2.70e+02	+	1.48e+03	2.24e+02	3.59e+03	3.33e+02	+	3.18e+03	3.54e+02	1.08e+04	7.68e+02	+	9.49e+03	7.14e+02
F11	5.04e+00	1.16e+01	≈	2.54e+00	2.09e+00	2.60e+01	3.56e+00	+	2.43e+01	3.75e+00	9.03e+01	3.16e+01	+	4.88e+01	2.96e+01
F12	1.58e+02	9.70e+01	≈	1.89e+02	1.29e+02	1.84e+03	4.77e+02	+	1.24e+03	3.93e+02	1.60e+04	7.14e+03	≈	1.83e+04	9.27e+03
F13	1.85e+01	5.84e+00	+	1.62e+01	3.41e+00	4.08e+01	2.99e+01	+	2.67e+01	1.76e+01	1.61e+02	4.39e+01	+	1.04e+02	3.49e+01
F14	2.16e+01	1.02e+00	+	2.13e+01	1.20e+00	2.41e+01	1.65e+00	≈	2.45e+01	2.96e+00	5.38e+01	7.59e+00	≈	5.18e+01	6.20e+00
F15	1.69e+00	1.08e+00	+	7.61e−01	5.59e−01	2.26e+01	2.47e+00	≈	2.20e+01	2.00e+00	1.64e+02	4.63e+01	+	1.24e+02	4.17e+01
F16	2.98e+01	4.67e+01	≈	5.12e+01	6.41e+01	3.99e+02	1.45e+02	≈	3.92e+02	1.35e+02	1.85e+03	3.65e+02	≈	1.69e+03	3.74e+02
F17	3.44e+01	7.31e+00	≈	3.35e+01	6.49e+00	3.01e+02	1.12e+02	≈	2.62e+02	8.45e+01	1.31e+03	2.81e+02	+	1.06e+03	2.46e+02
F18	1.96e+01	4.85e+00	≈	2.10e+01	5.27e−01	2.38e+01	1.29e+00	−	2.50e+01	1.61e+00	1.75e+02	3.42e+01	≈	1.72e+02	2.82e+01
F19	4.55e+00	1.44e+00	≈	5.03e+00	2.37e+00	1.28e+01	2.84e+00	+	1.17e+01	2.44e+00	9.51e+01	1.97e+01	+	6.25e+01	1.03e+01
F20	3.28e+01	5.06e+00	+	3.00e+01	7.19e+00	1.53e+02	7.11e+01	≈	1.42e+02	5.04e+01	1.48e+03	2.87e+02	+	1.30e+03	2.65e+02
F21	2.08e+02	2.33e+00	+	2.06e+02	1.69e+00	2.15e+02	3.93e+00	+	2.12e+02	2.93e+00	2.57e+02	8.03e+00	+	2.48e+02	6.00e+00
F22	1.00e+02	1.44e−14	≈	1.00e+02	1.44e−14	1.52e+03	1.95e+03	≈	2.24e+03	1.77e+3	1.17e+4	6.30e+2	+	1.03e+4	7.46e+2
F23	3.50e+02	3.06e+00	≈	3.49e+02	2.93e+00	4.32e+02	6.19e+00	+	4.27e+02	5.76e+00	5.68e+02	8.68e+00	≈	5.66e+02	9.49e+00
F24	4.26e+02	1.89e+00	≈	4.26e+02	2.21e+00	5.07e+02	3.62e+00	≈	5.08e+02	4.32e+00	9.02e+02	7.84e+00	+	8.97e+02	7.06e+00
F25	3.87e+02	7.06e−03	−	3.87e+02	1.30e−02	4.81e+02	2.75e+00	≈	4.81e+02	2.32e+00	7.32e+02	4.17e+01	+	6.99e+02	5.00e+01
F26	9.34e+02	4.56e+01	≈	9.43e+02	4.58e+01	1.14e+03	5.43e+01	≈	1.16e+03	6.16e+01	3.20e+03	7.38e+01	+	3.11e+03	6.92e+01
F27	4.97e+02	6.58e+00	+	4.94e+02	6.92e+00	5.10e+02	1.34e+01	+	5.02e+02	1.04e+01	5.84e+02	1.80e+01	+	5.76e+02	2.21e+01
F28	3.06e+02	2.54e+01	+	3.02e+02	1.60e+01	4.59e+02	1.29e−13	−	4.59e+02	2.15e−13	5.27e+02	2.12e+01	≈	5.29e+02	2.46e+01
F29	4.44e+02	1.01e+01	+	4.38e+02	8.73e+00	3.74e+02	1.51e+01	≈	3.69e+02	1.47e+01	1.25e+03	2.11e+02	+	1.14e+03	2.12e+02
F30	1.97e+03	1.08e+01	−	1.97e+03	8.58e+00	6.11e+05	4.54e+04	≈	5.98e+05	2.02e+04	2.34e+03	1.39e+02	≈	2.34e+03	1.59e+02
W	**13**					**12**					**21**				
L	**3**					**3**					**0**				
T	**13**					**14**					**8**				

Table 3
Overall performance comparisons of variant algorithms with their respective baseline algorithms according to Holm, Hochberg and Hommel procedures.

Algorithm	unadjusted p	p_{Holm}	$p_{Hochberg}$	p_{Hommel}
jSO	**0.000646**	**0.001292**	**0.001292**	**0.001292**
LSHADE–cnEpSin	**0.000029**	**0.000273**	**0.000258**	**0.000229**

global and local search, while avoiding the dilemma of easily falling into local optima caused by multi-function combinations. ADCC can well coordinate and balance the relationship between global and local search by leveraging competition. Dimensional fusion is able to address the problem of grouped optimization of sub-components. Meanwhile, the populations integrating the advantages of different strategies can facilitate the exploration of new regions and timely escape from local obstacles.

The multi-problem test results of the variant algorithms ADCC-jSO and ADCC-LSHADE-cnEpSin formed by applying the ADCC framework and their corresponding baseline algorithms are shown in Table 3. According to the Holm, Hochberg and Hommel procedures, it can be concluded that both variant algorithms are significantly superior to their respective baseline algorithms, with the obtained p-values all being significantly less than 0.05.

4.2. ADCC for high-level algorithm integration

To further verify the generality of the ADCC framework, it is extended to cross algorithm scenarios and the cooperation and competition framework is constructed by utilizing the inherent differences in the characteristics of different algorithms. The baseline algorithms jSO and LSHADE-cnEpSin are used as representatives. By comparing the Euclidean distance between the offspring generated by the two algorithms and the parent, the exploration and exploitation modules are defined, and the resulting algorithm is named ADCCDE.

The performance comparisons of 30-D, 50-D and 100-D functions are shown in Table 5. As can be seen from Table 5, our evaluations show that the overall performance of the proposed ADCCDE algorithm is better than the other two algorithms. Specifically, ADCCDE performs significantly better than jSO on 12, 19 and 25 functions, and loses on 2, 1 and 0 functions in the cases of 30-D, 50-D and 100-D respectively. It is worth noting that with an increase in dimensionality, the performance advantage of ADCC becomes more prominent. Compared with LSHADE-cnEpSin, ADCCDE is outstanding on 13, 18 and 16 functions and underperforms on 3, 5 and 4 functions respectively. The performance gap between the two is relatively uniform in the low, medium and high dimensions. In general, ADCCDE shows significant advantages in more than half of the 50-D and 100-D functions. These results show that ADCC can effectively deal with complex optimization problems, which shows the wide application of robustness and applicability.

Fig. 3 shows the convergence curves on nine selected functions. As seen, ADCCDE achieves the best final solutions on seven functions,

Table 4
Performance comparisons of ADCC–LSHADE–cnEpSin with LSHADE–cnEpSin on 30–D, 50–D and 100–D functions.

Problem	30–D					50–D					100–D				
	LSHADE–cnEpSin			ADCC–LSHADE–cnEpSin		LSHADE–cnEpSin			ADCC–LSHADE–cnEpSin		LSHADE–cnEpSin			ADCC–LSHADE–cnEpSin	
	mean	std	sig	mean	std	mean	std	sig	mean	std	mean	std	sig	mean	std
F1	0.00e+00	0.00e+00	≈	0.00e+00	0.00e+00	0.00e+00	0.00e+00	≈	0.00e+00	0.00e+00	0.00e+00	0.00e+00	≈	0.00e+00	0.00e+00
F3	0.00e+00	0.00e+00	≈	0.00e+00	0.00e+00	0.00e+00	0.00e+00	≈	0.00e+00	0.00e+00	0.00e+00	0.00e+00	≈	0.00e+00	0.00e+00
F4	4.14e+01	3.19e+00	–	4.40e+01	3.18e+00	4.72e+01	4.19e+01	–	5.24e+01	4.67e+01	2.01e+02	8.17e+00	≈	2.00e+02	8.43e+00
F5	1.23e+01	2.66e+00	+	9.45e+00	1.60e+00	2.59e+01	7.67e+00	+	1.66e+01	5.43e+00	5.52e+01	1.11e+01	+	3.43e+01	6.19e+00
F6	3.80e−08	7.74e−08	+	3.07e−09	1.07e−08	9.06e−07	7.93e−07	+	5.72e−07	6.12e−07	5.68e−05	1.55e−05	+	4.18e−05	1.80e−05
F7	4.31e+01	2.44e+00	+	3.92e+01	1.92e+00	7.48e+01	5.44e+00	+	6.63e+01	3.97e+00	1.63e+02	6.05e+00	+	1.42e+02	4.90e+00
F8	1.26e+01	2.34e+00	+	9.24e+00	1.93e+00	2.58e+01	6.38e+00	+	1.76e+01	4.83e+00	5.41e+01	9.34e+00	+	3.34e+01	5.98e+00
F9	0.00e+00	0.00e+00	≈	0.00e+00	0.00e+00	0.00e+00	0.00e+00	≈	0.00e+00	0.00e+00	0.00e+00	0.00e+00	≈	0.00e+00	0.00e+00
F10	1.44e+03	1.69e+02	≈	1.41e+03	2.09e+02	3.26e+03	2.56e+02	+	3.11e+03	3.56e+02	1.04e+04	4.88e+02	+	1.01e+04	5.91e+02
F11	1.54e+01	2.15e+01	≈	1.08e+01	1.79e+01	2.15e+01	1.96e+00	≈	2.22e+01	2.61e+00	5.44e+01	3.53e+01	≈	4.67e+01	2.90e+01
F12	3.96e+02	2.09e+02	≈	3.69e+02	1.82e+02	1.40e+03	3.46e+02	≈	1.34e+03	2.95e+02	4.59e+03	7.01e+02	≈	4.54e+03	6.13e+02
F13	1.65e+01	1.23e+01	≈	1.52e+01	6.58e+00	7.65e+01	3.63e+01	+	4.73e+01	3.27e+01	1.19e+02	3.99e+01	≈	1.03e+02	2.88e+01
F14	2.14e+01	3.62e+00	+	2.05e+01	4.04e+00	2.70e+01	2.48e+00	+	2.63e+01	2.94e+00	5.35e+01	8.20e+00	≈	5.20e+01	9.75e+00
F15	2.95e+00	1.82e+00	≈	3.03e+00	1.82e+00	2.62e+01	3.86e+00	+	2.43e+01	3.60e+00	9.82e+01	3.22e+01	+	8.14e+01	2.95e+01
F16	2.34e+01	3.21e+01	+	1.70e+01	2.30e+01	2.95e+02	1.09e+02	+	2.56e+02	1.14e+02	1.18e+03	2.82e+02	+	9.43e+02	2.84e+02
F17	2.96e+01	6.52e+00	+	2.71e+01	5.84e+00	2.35e+02	6.67e+01	+	2.12e+02	6.85e+01	9.70e+02	1.95e+02	+	8.26e+02	1.54e+02
F18	2.13e+01	7.73e−01	≈	2.12e+01	8.07e−01	2.45e+01	2.03e+00	≈	2.43e+01	1.98e+00	7.35e+01	1.84e+01	≈	6.97e+01	1.53e+01
F19	5.55e+00	1.63e+00	≈	5.77e+00	1.67e+00	1.73e+01	2.71e+00	≈	1.69e+01	3.49e+00	5.68e+01	9.02e+00	≈	5.54e+01	5.51e+00
F20	3.11e+01	6.68e+00	≈	3.10e+01	7.87e+00	1.13e+02	3.25e+01	≈	1.03e+02	1.54e+01	1.08e+03	1.96e+02	≈	1.05e+03	1.37e+02
F21	2.12e+02	2.77e+00	+	2.09e+02	2.46e+00	2.27e+02	7.10e+00	+	2.19e+02	5.86e+00	2.78e+02	9.90e+00	+	2.59e+02	5.93e+00
F22	1.00e+02	1.44e−14	≈	1.00e+02	6.39e−14	1.18e+03	1.56e+03	≈	1.24e+03	1.63e+03	1.06e+04	6.78e+02	+	9.83e+03	7.27e+02
F23	3.56e+02	3.39e+00	+	3.52e+02	3.83e+00	4.40e+02	7.11e+00	+	4.31e+02	5.72e+00	5.95e+02	9.36e+00	+	5.74e+02	9.16e+00
F24	4.29e+02	2.46e+00	+	4.27e+02	2.19e+00	5.13e+02	6.14e+00	+	5.10e+02	4.34e+00	9.17e+02	1.50e+01	+	9.00e+02	9.66e+00
F25	3.87e+02	6.20e−03	–	3.87e+02	7.00e−03	4.80e+02	4.06e−01	–	4.81e+02	2.86e+00	6.91e+02	4.41e+01	≈	6.92e+02	4.69e+01
F26	9.55e+02	4.62e+01	+	9.01e+02	4.33e+01	1.20e+03	1.04e+02	+	1.12e+03	7.69e+01	3.08e+03	1.13e+02	+	3.01e+03	6.39e+01
F27	5.04e+02	6.29e+00	≈	5.05e+02	5.94e+00	5.29e+02	1.87e+01	≈	5.29e+02	1.56e+01	5.90e+02	1.86e+01	≈	5.88e+02	1.56e+01
F28	3.13e+02	3.65e+01	≈	3.21e+02	4.36e+01	4.59e+02	1.17e+01	–	4.60e+02	1.17e+01	5.14e+02	1.79e+01	≈	5.12e+02	2.04e+01
F29	4.36e+02	6.32e+00	≈	4.34e+02	6.37e+00	3.55e+02	1.07e+01	+	3.50e+02	8.78e+00	1.17e+03	1.54e+02	+	1.05e+03	1.23e+02
F30	2.00e+03	7.32e+01	≈	2.00e+03	7.21e+01	6.48e+05	6.36e+04	≈	6.47e+05	8.35e+04	2.38e+03	1.72e+02	≈	2.38e+03	1.96e+02
W	**11**					**15**					**14**				
L	**2**					**3**					**0**				
T	**16**					**11**					**15**				

including 30-D F8, 50-D F6, 50-D F8, 50-D F26, 100-D F8, 100-D F20,100-D F22. ADCCDE and LSHADE–cnEpSin achieve comparable results in 30-D F16. On the 100-D F23, the convergence speed of ADCCDE is slightly lower than that of jSO in the mid stage, but remains comparable to jSO in the early and late stages, and the final error value is comparable. This short-term slow convergence may be attributed to the fact that ADCCDE maintains its own population diversity to avoid falling into local optima. It can be seen that after dynamic adjustment, its convergence gradually catches up with that of jSO. It is observable that for multimodal problems such as 50-D and 100-D F8, there is a temporary slow convergence period in the middle stage of iteration. This period preserves the diversity at the initial iteration stage, thus laying the groundwork for subsequent catching-up. However, it is undeniable that the late-stage local optima escape effect brought by this short-term slow convergence is excellent, and the final convergence performance is more superior.

Furthermore, Table 6 presents the p-values obtained from the multi-problem test. ADCCDE statistically outperforms jSO and LSHADE-cnEpSin, with all p-values being significantly lower than 0.05. Overall, compared to the two baseline algorithms, ADCCDE exhibits significantly superior performance.

To investigate the time complexity of ADCCDE, the evaluation methodology proposed in Awad et al. (2016) is adopted. The constituent algorithm jSO and LSHADE-cnEpSin are selected as comparison baselines. This methodology requires measuring three time components, namely T_0, T_1, and T_2, in order to eliminate extraneous influences on the measured computational cost. The details are described as follows.

Since different computing platforms exhibit heterogeneous CPU and memory performance, T_0 is introduced to eliminate the impact of the external computational environment. Specifically, T_0 is defined as the execution time of the following benchmark program:

$$
\begin{aligned}
&for \quad i = 1 : 1,000,000\\
&\quad x = 0.55 + (double)\ i;\\
&\quad x = x + x; x = x/2; x = x * x; x = sqrt(x); x = log(x);\\
&\quad x = exp(x); x = x/(x + 2);\\
&end
\end{aligned}
$$

T_1 is defined as the time consumed by performing 200,000 function evaluations on the F18 benchmark problem. The purpose is to eliminate the inherent computational cost of the test function itself from the overall runtime of the algorithm. T_2 corresponds to the execution time of the F18 function under 200,000 evaluations. To reduce uncertainty caused by stochastic runtime fluctuations, T_2 is calculated as the average over five independent runs. $(T_2 - T_1)/T_0$ is used to measure the final time complexity of the algorithm, and the experiments are conducted on 30-D, 50-D, and 100-D problems.

The experimental results are summarized in Table 7. The results indicate that jSO exhibits lower computational complexity in the 30-D, 50-D, and 100-D problems, and a relatively large complexity difference between ADCCDE and LSHADE-cnEpSin is observed in the 30-D and 100-D. The reason for the high complexity of ADCCDE may be that our competition is a macroscopic competition mode for the population, so these two sub-algorithms integrated into the algorithm need to run their own offspring each generation, which increases the time complexity to a certain extent.

Table 5
Performance comparisons of ADCCDE with the baselines on 30–D, 50–D and 100–D functions over 51 independent runs.

Problem	30-D					50-D					100-D				
	jSO		LSHADE–cnEpSin		ADCCDE	jSO		LSHADE–cnEpSin		ADCCDE	jSO		LSHADE–cnEpSin		ADCCDE
	mean(std)	sig	mean(std)	sig	mean(std)	mean(std)	sig	mean(std)	sig	mean(std)	mean(std)	sig	mean(std)	sig	mean(std)
F1	0.00e+00(0.00e+00)	≈	0.00e+00(0.00e+00)	≈	0.00e+00(0.00e+00)	0.00e+00(0.00e+00)	≈	0.00e+00(0.00e+00)	≈	0.00e+00(0.00e+00)	0.00e+00(0.00e+00)	≈	0.00e+00(0.00e+00)	≈	0.00e+00(0.00e+00)
F3	0.00e+00(0.00e+00)	≈	0.00e+00(0.00e+00)	≈	0.00e+00(0.00e+00)	0.00e+00(0.00e+00)	≈	0.00e+00(0.00e+00)	≈	0.00e+00(0.00e+00)	3.49e−06(2.68e−06)	+	0.00e+00(0.00e+00)	−	8.12e−08(8.24e−08)
F4	5.86e+01(0.00e+00)	+	4.14e+01(3.19e+00)	−	4.90e+01(2.16e+00)	5.85e+01(5.04e+01)	+	4.72e+01(4.19e+01)	−	5.14e+01(4.44e+01)	1.99e+02(1.52e+01)	+	2.01e+02(8.17e+00)	≈	1.99e+02(1.44e+01)
F5	7.43e+00(2.29e+00)	≈	1.23e+01(2.66e+00)	+	7.75e+00(1.47e+00)	1.36e+01(3.31e+00)	≈	2.59e+01(7.67e+00)	+	1.41e+01(2.17e+00)	3.67e+01(8.37e+00)	+	5.52e+01(1.11e+01)	+	2.76e+01(3.35e+00)
F6	1.07e−08(3.72e−08)	+	3.80e−08(7.74e−08)	+	0.00e+00(0.00e+00)	2.29e−07(4.06e−07)	+	9.06e−07(7.93e−07)	+	5.77e−08(1.09e−07)	7.69e−05(1.25e−04)	+	5.68e−05(1.55e−05)	+	1.79e−06(1.27e−06)
F7	3.83e+01(2.04e+00)	≈	4.31e+01(2.44e+00)	+	3.89e+01(1.48e+00)	6.67e+01(3.43e+00)	+	7.48e+01(5.44e+00)	+	6.47e+01(2.19e+00)	1.44e+02(8.67e+00)	+	1.63e+02(6.05e+00)	+	1.27e+02(3.22e+00)
F8	8.17e+00(1.98e+00)	≈	1.26e+01(2.34e+00)	+	8.24e+00(1.44e+00)	1.52e+01(2.92e+00)	+	2.58e+01(6.38e+00)	+	1.34e+01(1.81e+00)	3.68e+01(8.43e+00)	+	5.41e+01(9.34e+00)	+	2.92e+01(4.88e+00)
F9	0.00e+00(0.00e+00)	≈	0.00e+00(0.00e+00)	≈	0.00e+00(0.00e+00)	0.00e+00(0.00e+00)	≈	0.00e+00(0.00e+00)	≈	0.00e+00(0.00e+00)	1.94e−02(7.96e−02)	+	0.00e+00(0.00e+00)	≈	0.00e+00(0.00e+00)
F10	1.69e+03(2.70e+02)	+	1.44e+03(1.69e+02)	≈	1.47e+03(2.20e+02)	3.59e+03(3.33e+02)	+	3.26e+03(2.56e+02)	+	3.11e+03(3.05e+02)	1.08e+04(7.68e+02)	+	1.04e+04(4.88e+02)	+	8.97e+03(4.65e+02)
F11	5.04e+00(1.16e+01)	≈	1.54e+01(2.15e+01)	+	8.94e+00(1.73e+01)	2.60e+01(3.56e+00)	+	2.15e+01(1.96e+00)	+	2.05e+01(2.13e+00)	9.03e+01(3.16e+01)	+	5.44e+01(3.53e+01)	≈	4.59e+01(3.54e+01)
F12	1.58e+02(9.70e+01)	≈	3.96e+02(2.09e+02)	+	1.35e+02(8.33e+01)	1.84e+03(4.77e+02)	+	1.40e+03(3.46e+02)	+	8.68e+02(3.92e+02)	1.60e+04(7.14e+03)	+	4.59e+03(7.01e+02)	≈	4.92e+03(1.03e+03)
F13	1.85e+01(5.84e+00)	+	1.65e+01(1.23e+01)	−	1.74e+01(3.99e+00)	4.08e+01(2.99e+01)	+	7.65e+01(3.63e+01)	+	3.48e+01(2.81e+01)	1.61e+02(4.39e+01)	+	1.19e+02(3.99e+01)	+	4.65e+01(2.37e+01)
F14	2.16e+01(1.02e+00)	≈	2.14e+01(3.62e+00)	+	2.10e+01(2.70e+00)	2.41e+01(1.65e+00)	+	2.70e+01(2.48e+00)	+	2.27e+01(1.15e+00)	5.38e+01(7.59e+00)	+	5.35e+01(8.20e+00)	+	3.75e+01(4.04e+00)
F15	1.69e+00(1.08e+00)	+	2.95e+00(1.82e+00)	+	9.79e−01(6.64e−01)	2.26e+01(2.47e+00)	+	2.62e+01(3.86e+00)	+	2.02e+01(1.53e+00)	1.64e+02(4.63e+01)	+	9.82e+01(3.22e+01)	+	5.39e+01(3.14e+01)
F16	2.98e+01(4.67e+01)	+	2.34e+01(3.21e+01)	≈	3.14e+01(4.70e+01)	3.99e+02(1.45e+02)	+	2.95e+02(1.09e+02)	≈	3.16e+02(1.24e+02)	1.85e+03(3.65e+02)	+	1.18e+03(2.82e+02)	−	1.42e+03(2.56e+02)
F17	3.44e+01(7.31e+00)	+	2.96e+01(6.52e+00)	≈	3.13e+01(6.21e+00)	3.01e+02(1.12e+02)	+	2.35e+02(6.67e+01)	+	1.95e+02(7.66e+01)	1.31e+03(2.81e+02)	+	9.70e+2(1.95e+2)	−	1.07e+3(1.66e+2)
F18	1.96e+01(4.85e+00)	+	2.13e+01(7.73e−01)	+	2.06e+01(1.99e−01)	2.38e+01(1.29e+00)	+	2.45e+01(2.03e+00)	+	2.23e+01(1.26e+00)	1.75e+02(3.42e+01)	+	7.35e+01(1.84e+01)	+	4.58e+01(1.04e+01)
F19	4.55e+00(1.44e+00)	−	5.55e+00(1.63e+00)	≈	5.47e+00(1.77e+00)	1.28e+01(2.84e+00)	≈	1.73e+01(2.71e+00)	+	1.33e+01(2.55e+00)	9.51e+01(1.97e+01)	+	5.68e+01(9.02e+00)	+	4.51e+01(4.97e+00)
F20	3.28e+01(5.06e+00)	+	3.11e+01(6.68e+00)	≈	2.90e+01(7.20e+00)	1.53e+02(7.11e+01)	+	1.13e+02(3.25e+01)	≈	1.19e+02(4.22e+01)	1.48e+03(2.87e+02)	+	1.08e+03(1.96e+02)	≈	1.15e+03(1.82e+02)
F21	2.08e+02(2.33e+00)	≈	2.12e+02(2.77e+00)	+	2.08e+02(1.65e+00)	2.15e+02(3.93e+00)	≈	2.27e+02(7.10e+00)	+	2.15e+02(2.40e+00)	2.57e+02(8.03e+00)	+	2.78e+02(9.90e+00)	+	2.52e+02(4.27e+00)
F22	1.00e+02(1.44e−14)	≈	1.00e+02(1.44e−14)	≈	1.00e+02(1.44e−14)	1.52e+03(1.95e+03)	≈	1.18e+03(1.56e+03)	−	2.85e+03(1.49e+03)	1.17e+04(6.30e+02)	+	1.06e+04(6.78e+02)	+	9.79e+03(5.04e+02)
F23	3.50e+02(3.06e+00)	≈	3.56e+02(3.39e+00)	+	3.50e+02(2.64e+00)	4.32e+2(6.19e+00)	+	4.40e+02(7.11e+00)	+	4.28e+02(4.72e+00)	5.68e+02(8.68e+00)	+	5.95e+2(9.36e+00)	+	5.56e+02(1.13e+01)
F24	4.26e+02(1.89e+00)	≈	4.29e+02(2.46e+00)	+	4.26e+02(1.92e+00)	5.07e+02(3.62e+00)	≈	5.13e+02(6.14e+00)	+	5.08e+02(3.06e+00)	9.02e+02(7.84e+00)	≈	9.17e+02(1.50e+01)	+	8.99e+02(5.69e+00)
F25	3.87e+02(7.06e−03)	+	3.87e+02(6.20e−03)	−	3.87e+02(6.30e−03)	4.81e+02(2.75e+00)	+	4.80e+02(4.06e−01)	−	4.81e+02(2.62e+00)	7.32e+02(4.17e+01)	+	6.91e+02(4.41e+01)	≈	6.86e+02(4.82e+01)
F26	9.34e+02(4.56e+01)	+	9.55e+02(4.62e+01)	+	9.04e+02(4.10e+01)	1.14e+03(5.43e+01)	+	1.20e+03(1.04e+02)	+	1.09e+03(5.51e+01)	3.20e+03(7.38e+01)	+	3.08e+03(1.13e+02)	≈	3.12e+03(1.01e+02)
F27	4.97e+02(6.58e+00)	−	5.04e+02(6.29e+00)	≈	5.02e+02(6.88e+00)	5.10e+02(1.34e+01)	−	5.29e+02(1.87e+01)	+	5.23e+02(1.78e+01)	5.84e+2(1.80e+01)	≈	5.90e+02(1.86e+1)	+	5.76e+02(2.15e+01)
F28	3.06e+02(2.54e+01)	≈	3.13e+02(3.65e+01)	≈	3.17e+02(4.00e+01)	4.59e+02(1.29e−13)	+	4.59e+02(1.17e+01)	−	4.60e+02(9.47e+00)	5.27e+02(2.12e+01)	+	5.14e+02(1.79e+01)	−	5.19e+02(2.47e+01)
F29	4.44e+02(1.01e+01)	+	4.36e+02(6.32e+00)	≈	4.37e+02(7.42e+00)	3.74e+2(1.51e+1)	+	3.55e+02(1.07e+01)	−	3.62e+02(1.08e+01)	1.25e+03(2.11e+02)	≈	1.17e+03(1.54e+02)	≈	1.21e+03(1.67e+02)
F30	1.97e+03(1.08e+01)	≈	2.00e+03(7.32e+01)	≈	1.97e+03(1.22e+01)	6.11e+05(4.54e+04)	≈	6.48e+05(6.36e+04)	≈	6.42e+05(6.51e+04)	2.34e+03(1.39e+02)	+	2.38e+03(1.72e+02)	+	2.29e+03(1.70e+02)
W	**12**		**13**			**19**		**18**			**25**		**16**		
L	**2**		**3**			**1**		**5**			**0**		**4**		
T	**15**		**13**			**9**		**6**			**4**		**9**		

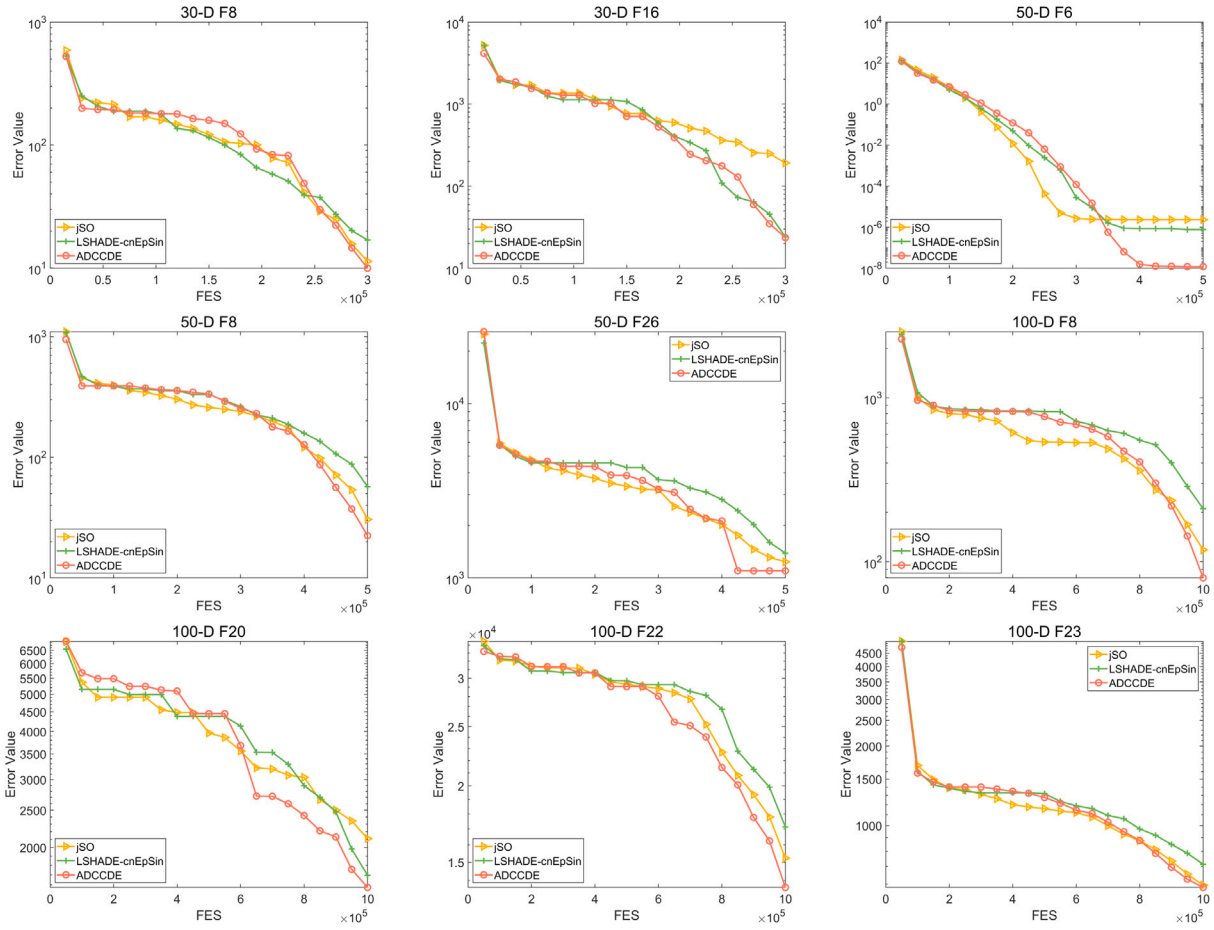


Fig. 3. Convergence plot of the compared DE algorithms on nine selected functions.

Table 6
Overall performance comparisons of ADCCDE with the baselines according to Holm, Hochberg and Hommel procedures.

Algorithm	unadjusted p	p_{Holm}	$p_{Hochberg}$	p_{Hommel}
jSO	**0.000006**	**0.000013**	**0.000013**	**0.000013**
LSHADE–cnEpSin	**0.000022**	**0.000022**	**0.000022**	**0.000022**

Table 7
Time complexity of the high-level integrated framework and its constituent algorithms (T_0, T_1, T_2 in seconds).

Dimension	Algorithm	T_0	T_1	T_2	$(T_2 - T_1)/T_0$
30-D	jSO	0.01	0.55	1.84	30
	LSHADE-cnEpSin			1.85	129
	ADCCDE			2.86	231
50-D	jSO	0.01	1.11	1.48	37
	LSHADE-cnEpSin			3.25	214
	ADCCDE			3.28	217
100-D	jSO	0.01	3.62	4.13	51
	LSHADE-cnEpSin			5.87	225
	ADCCDE			7.19	357

4.3. Comparison with state-of-the-art DE variants

To demonstrate the performance of ADCCDE, it is further compared with the following eleven state-of-the-art DE variants:

Adaptative Differential Evolution With Diversity Maintenance And Restart Mechanism (*ADE-DMRM*) (Lin and Meng, 2024): An adaptive DE algorithm integrating diversity maintenance and dimension-learning restart mechanism, utilizing wavelet basis functions and

Cauchy distributions for scale factor generation, and optimizing external archive based on successful evolution rate to enhance global search capability.

Selective-Candidate Framework With Similarity Selection Rule L-SHADE (*SCSS-L-SHADE*) (Zhang et al., 2020): A selective-candidate framework for L-SHADE, utilizing similarity selection rules to generate multiple candidates and balance exploitation–exploration via fitness ranking and solution-space distance.

Fractional Order DE (*FODE*) (Wang et al., 2024): A fractional-order differential evolution algorithm integrating a bi-strategy co-deployment framework and fractional calculus-enhanced differential vectors, utilizing historical population information to boost exploration ability and maintain population diversity for superior optimization performance in complex problems.

LSHADE Algorithm With A Rank-Based Selective Pressure Strategy (*LSHADE-RSP*) (Stanovov et al., 2018): An enhanced LSHADE algorithm with rank-based selective pressure strategy, featuring adaptive mutation weights and parameter adaptation to boost optimization performance in complex multimodal problems.

Explicitly Adaptive DE (*EaDE*) (Zhang et al., 2021): An adaptive DE algorithm with explicit exploitation–exploration controls, featuring stage—wise strategy switching and evolution difficulty detection to optimize performance in dynamic search landscapes.

Domain Transform DE (*DTDE*) (Zhang et al., 2023): An enhanced DE algorithm that transforms the fitness landscape to the frequency domain, removes high-frequency components, and uses an evolution difficulty-based mechanism to address premature convergence and stagnation.

DE With Exponential Crossover (*DE-EXP*) (Meng and Chen, 2023): An enhanced DE algorithm with exponential crossover and fitness-value-independent parameter control, featuring automatically generated crossover rate *Cr* and population size reduction strategy to boost optimization performance.

Multiselection-Based DE (*MSDE*) (Cai et al., 2024): A differential evolution variant employing a multiselection strategy that orderly activates one-to-one, fitness-based, and distance-based selections based on stagnation detection, gradually shrinking the population to balance exploration and exploitation, thereby enhancing optimization performance in complex problems with numerous local optima or high dimensions.

Dynamic Fitness-Distance Balance-Based LSHADE (*dFDB-LSHADE*) (Yildirim et al., 2025): An algorithm based on LSHADE incorporating dynamic fitness-distance balance to guide the selection of optimal individuals in the population.

Modified LSHADE With Semi-Parameter Adaptation Hybrid With CMA-ES (*mLSHADE-SPACMA*) (Fu et al., 2025): An improved LSHADE-SPACMA framework integrating a precise elimination and generation mechanism, modified SPACMA semi-parametric adaptation, a rank-based selective pressure mutation strategy, and an elite external archive mechanism.

Natural Survivor Method-Based LSHADE-CnEpSin (*NSM-LSHADE-CnEpSin*) (Öztürk and Kahraman, 2025): An Advanced LSHADE-CnEpSin framework that reconstructs the update mechanism based on the Natural Survivor Method and balances population diversity and local exploitation capability through dynamic reference signals.

Performance comparisons on 30-D, 50-D and 100-D functions are shown in Tables 8–10. The data highlighted with a gray background represent that the average fitness value of the algorithm is the best among all the compared algorithms. The final statistical comparison results are displayed in the last row of the table. From the summary results shown in Table 11, ADCCDE performs overall better than most of the competitors.

Specifically, in 30-D, compared with LSHADE-RSP, SCSS -L-SHADE, DE-EXP, dFDB-LSHADE, mLSAHDE-SPACMA and NSM-LSHADE-CnEpSin, ADCCDE has significant advantages, with the "W/L" ratio reaching twice or even more. However, it is inferior to FODE, DTDE and MSDE, performing better only on 4, 7 and 7 functions respectively. It performs basically on a par with the remaining two algorithms EaDE and ADE-DMRM. ADCCDE has greater advantages on functions F12, F15, F25 and F26. In low dimensions, it has high adaptability to medium and low complexity scenarios. But it has relatively low advantages on F29.

With the increase of dimensions, the ADCCDE algorithm exhibits significant advantages. In 100-D, it shows overwhelming advantages over 8 algorithms. In 50-D, it still has obvious advantages over the algorithms LSHADE-RSP, SCSS-L-SHADE, DE-EXP, dFDB-LSHADE, mLSAHDE-SPACMA and NSM-LSHADE-CnEpSin which it outperformed by a large margin in 30-D. The "W/L" ratios climb to "11/4", "14/5", "17/8", "17/5", "14/6" and "20/2" respectively. In 100-D, except for mLSAHDE-SPACMA and NSM-LSHADE-CnEpSin, the "W/L" ratios of these algorithms are further increased to "21/1",“ 22/1" and "20/4". There is even no loss against dFDB-LSHADE.

For algorithms such as EaDE and ADE-DMRM, which had comparable performance on functions in 30-D, ADCCDE gradually surpasses them by a large margin in 50-D and 100-D. Moreover, for the two algorithms DTDE and MSDE that outperformed ADCCDE in 30-D and 50-D, they are also surpassed by ADCCDE in the high-dimensional 100-D scenario, and the performance is basically on a par. As for FODE, it defeated ADCCDE in 30-D, drew with ADCCDE in 50-D, and was then far surpassed by ADCCDE in 100-D. However, ADCCDE still performs on a par with mLSAHDE-SPACMA, which has good high-dimensional performance. For functions in 50-D and 100-D, ADCCDE continues its advantages on functions F12, F15 and F25 and still performs well. This indicates that in high dimensions, the algorithm still has high adaptability to medium and low complexity scenarios. It can be seen that for the hybrid functions F11–F20 and composition functions F21–F28 corresponding to such complexity scenarios, the winning ratio increases with the rise of dimensions. Nevertheless, it has to be admitted that its performance is indeed deficient on F29, a high-order composition function with high complexity.

Fig. 4 depicts the convergence curves of ADCCDE and eleven state-of-the-art DEs on nine functions. It can be seen that ADCCDE achieves the minimum values on five functions. Compared with the other eleven algorithms, the final convergence effect of ADCCDE is significantly better than that of other advanced algorithms on 100-D F13, F15 and F18. Moreover, it basically achieves the optimal values together with mLSAHDE-SPACMA on 50-D F12 and 100-D F11. On 30-D F12, MSDE has the fastest convergence rate at the initial stage, but it is surpassed by ADCCDE in the later stage. The final error values of ADCCDE on 30-D F4 and 50-D F4 are worse than those of MSDE and FODE, while the error between ADCCDE and the optimal values of multiple algorithms is not significant on 50-D F18.

On functions such as 30-D F12, 100-D F11, 100-D F15 and 100-D F18, ADCCDE exhibits a phenomenon of slow convergence in the middle stage compared with other advanced algorithms. This is also because it intends to maintain the diversity of its population in the middle stage. At this time, the evolution scale is small, and an exploratory population is selected to maintain the exploration of a broad space, thus achieving transcendence in the later stage. It can be observed that these functions all have a turning point where convergence accelerates in the later stage.

To draw statistical conclusions, Table 12 shows the Friedman test (Cleophas and Zwinderman, 2016) ranking of twelve algorithms. The rankings of FODE, ESADE, DTDE, MSDE and mLSHADE-SPACMA are close. The average ranking of ADCCDE is 6.253, better than the five algorithms mentioned above, and significantly superior to other advanced algorithms.

4.4. Comparison between ADCC and ESA framework

To verify the effectiveness of dual module fusion, we first compared the ADCCDE algorithm described in Section 3 with the ESADE algorithm. ESADE implemented dynamic selection of offspring generated

Table 8
Performance comparison of with state-of-the-art DEs on 30–D functions.

Problem	ADE-DMRM	SCSS-L-SHADE	FODE	LSHADE-RSP	EaDE	DTDE	DE-EXP	MSDE	dFDB-LSHADE	mLSHADE-SPACMA	NSM-LSHADE-CnEpSin	ADCCDE
F1	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0)
F3	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0)
F4	5.48e+1 (1.63e+1) +	5.86e+1 (2.75e−14) +	2.24e+1 (3.51e+0) −	5.86e+1 (1.39e−14) +	5.86e+1 (4.39e−13) +	5.86e+1 (2.52e−14) +	5.88e+1 (1.09e+0) +	2.34e+1 (1.11e+0) −	5.86e+1 (2.69e−14) +	5.86e+1 (8.04e−15) +	5.74e+1 (8.20e+0) +	4.90e+1 (2.16e+0)
F5	6.87e+0 (1.40e+0) −	7.19e+0 (1.38e+0) −	6.95e+0 (1.75e+0) −	7.20e+0 (2.07e+0) ≈	3.63e+0 (2.24e+0) −	1.27e+0 (9.97e−1) −	1.05e+1 (2.95e+0) +	1.10e+1 (5.37e+0) +	6.40e+0 (1.33e+0) −	3.98e+0 (1.97e+0) −	1.24e+1 (2.17e+0) +	7.75e+0 (1.47e+0)
F6	1.11e−13 (1.59e−14) +	0.00e+0 (0.00e+0) ≈	1.34e−8 (4.11e−8) +	1.14e−8 (3.73e−8) +	0.00e+0 (0.00e+0) ≈	5.37e−9 (2.68e−8) ≈	1.28e−8 (3.75e−8) +	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	1.07e−8 (3.72e−8) +	0.00e+0 (0.00e+0)
F7	3.69e+1 (1.20e+0) −	3.77e+1 (1.72e+0) −	3.78e+1 (1.82e+0) −	3.87e+1 (1.60e+0) ≈	3.65e+1 (2.27e+0) −	3.40e+1 (9.39e−1) −	4.18e+1 (2.97e+0) +	4.03e+1 (4.72e+0) ≈	3.74e+1 (1.45e+0) −	3.42e+1 (9.38e−1) −	4.32e+1 (2.55e+0) +	3.89e+1 (1.48e+0)
F8	7.56e+0 (1.41e+0) ≈	7.63e+0 (1.69e+0) −	6.81e+0 (2.19e+0) −	7.35e+0 (1.94e+0) −	4.41e+0 (2.36e+0) −	1.35e+0 (9.91e−1) −	1.05e+1 (2.88e+0) +	1.04e+1 (4.20e+0) ≈	6.98e+0 (1.25e+0) −	3.93e+0 (1.86e+0) −	1.41e+1 (2.25e+0) +	8.24e+0 (1.44e+0)
F9	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0)
F10	1.43e+3 (2.26e+2) ≈	1.42e+3 (2.19e+2) ≈	1.38e+3 (3.09e+2) ≈	1.56e+3 (2.58e+2) +	1.45e+3 (3.17e+2) ≈	4.92e+1 (8.75e+1) −	1.40e+3 (1.87e+2) ≈	9.57e+2 (5.84e+2) −	1.45e+3 (1.98e+2) ≈	1.40e+3 (3.32e+2) ≈	1.50e+3 (2.44e+2) ≈	1.47e+3 (2.20e+2)
F11	8.56e+0 (1.42e+1) +	1.65e+1 (2.28e+1) +	3.58e+0 (4.83e+0) −	1.24e+1 (2.11e+1) ≈	2.37e+1 (2.76e+1) +	2.32e+1 (2.71e+1) +	4.47e+0 (5.69e+0) ≈	5.26e+0 (2.87e+0) +	1.96e+1 (2.55e+1) +	1.27e+1 (2.13e+1) ≈	5.03e+0 (8.14e+0) ≈	8.94e+0 (1.73e+1)
F12	2.87e+2 (1.67e+2) +	6.08e+2 (2.41e+2) +	3.68e+2 (2.09e+2) +	1.08e+2 (8.96e+1) −	5.67e+2 (2.87e+2) +	5.25e+2 (2.62e+2) +	4.20e+2 (1.86e+2) +	3.93e+2 (2.25e+2) +	9.92e+2 (3.29e+2) +	2.13e+2 (1.56e+2) +	3.79e+2 (1.94e+2) +	1.35e+2 (8.33e+1)
F13	1.54e+1 (1.25e+1) −	1.49e+1 (5.91e+0) −	1.43e+1 (6.73e+0) −	1.87e+1 (3.56e+0) ≈	1.75e+1 (3.75e+0) ≈	1.79e+1 (5.23e+0) ≈	1.32e+1 (6.69e+0) −	1.67e+1 (5.49e+0) ≈	1.56e+1 (5.86e+0) ≈	1.30e+1 (5.59e+0) −	1.62e+1 (6.69e+0) ≈	1.74e+1 (3.99e+0)
F14	2.34e+1 (3.05e+0) +	2.19e+1 (2.99e+0) +	1.84e+1 (7.58e+0) ≈	2.04e+1 (4.79e+0) ≈	2.13e+1 (5.29e+0) +	1.95e+1 (8.75e+0) +	1.50e+1 (8.47e+0) ≈	2.10e+1 (5.60e+0) ≈	2.22e+1 (1.62e+0) +	2.16e+1 (1.05e+0) ≈	2.02e+1 (5.70e+0) ≈	2.10e+1 (2.70e+0)
F15	4.30e+0 (2.62e+0) +	1.81e+0 (9.74e−1) +	1.25e+0 (1.13e+0) ≈	1.15e+0 (7.25e−1) +	2.28e+0 (1.43e+0) +	5.08e+0 (2.05e+0) +	3.46e+0 (1.84e+0) +	1.53e+0 (1.11e+0) ≈	2.78e+0 (1.59e+0) +	1.81e+0 (1.21e+0) +	4.09e+0 (1.85e+0) +	9.79e−1 (6.64e−1)
F16	7.73e+1 (8.23e+1) +	3.99e+1 (4.10e+1) +	1.65e+1 (2.48e+1) −	2.62e+1 (3.39e+1) ≈	3.45e+1 (4.19e+1) +	1.41e+1 (1.09e+0) −	1.16e+2 (9.29e+1) +	2.56e+1 (4.27e+1) −	4.98e+1 (4.76e+1) +	3.77e+1 (5.11e+1) +	1.99e+1 (2.01e+1) ≈	3.14e+1 (4.70e+1)
F17	2.57e+1 (6.49e+0) −	3.45e+1 (5.37e+0) +	2.85e+1 (1.00e+1) ≈	3.55e+1 (6.44e+0) +	3.39e+1 (8.03e+0) ≈	2.06e+1 (5.56e+0) −	2.51e+1 (8.96e+0) −	2.65e+1 (3.37e+0) −	3.28e+1 (5.55e+0) ≈	2.95e+1 (6.38e+0) ≈	3.57e+1 (5.31e+0) +	3.13e+1 (6.21e+0)
F18	2.11e+1 (7.64e−1) +	2.10e+1 (7.07e−1) +	2.06e+1 (3.28e−1) ≈	2.08e+1 (3.36e−1) +	2.05e+1 (2.91e+0) +	2.61e+1 (2.57e+0) +	2.09e+1 (2.11e+0) +	2.01e+1 (4.06e+0) ≈	2.16e+1 (1.05e+0) +	2.16e+1 (8.41e−1) +	2.06e+1 (2.98e+0) ≈	2.06e+1 (1.99e−1)
F19	6.04e+0 (1.69e+0) ≈	5.07e+0 (1.85e+0) ≈	3.43e+0 (1.08e+0) −	3.64e+0 (1.16e+0) −	5.65e+0 (1.82e+0) ≈	4.99e+0 (1.55e+0) ≈	5.00e+0 (1.45e+0) ≈	3.69e+0 (1.19e+0) −	5.29e+0 (1.64e+0) ≈	5.99e+0 (1.87e+0) ≈	5.39e+0 (1.57e+0) ≈	5.47e+0 (1.77e+0)
F20	3.27e+1 (5.71e+0) +	3.09e+1 (5.63e+0) ≈	2.56e+1 (8.05e+0) −	3.16e+1 (4.92e+0) ≈	2.98e+1 (8.60e+0) ≈	1.80e+1 (7.87e+0) −	2.34e+1 (9.73e+0) −	2.49e+1 (2.51e+1) −	3.20e+1 (6.71e+0) +	6.92e+1 (4.73e+1) +	4.59e+1 (9.81e+0) +	2.90e+1 (7.20e+0)
F21	2.07e+2 (1.55e+0) −	2.07e+2 (1.59e+0) ≈	2.07e+2 (1.76e+0) −	2.08e+2 (1.71e+0) ≈	2.06e+2 (1.73e+0) −	2.02e+2 (1.80e+0) −	2.11e+2 (2.96e+0) +	2.10e+2 (3.98e+0) +	2.07e+2 (1.55e+0) ≈	2.09e+2 (4.71e+0) ≈	2.12e+2 (3.10e+0) +	2.08e+2 (1.65e+0)
F22	1.00e+2 (6.39e−14) −	1.00e+2 (1.44e−14) ≈	1.00e+2 (0.00e+0) −	1.00e+2 (1.44e−14) ≈	1.00e+2 (1.48e−13) −	1.00e+2 (6.39e−14) −	1.00e+2 (1.44e−14) ≈	1.00e+2 (0.00e+0) −	1.00e+2 (1.44e−14) ≈	1.00e+2 (1.90e−13) −	1.00e+2 (1.44e−14) ≈	1.00e+2 (1.44e−14)
F23	3.46e+2 (5.16e+0 −	3.50e+2 (3.10e+0) ≈	3.50e+2 (3.14e+0) ≈	3.51e+2 (3.26e+0) ≈	3.50e+2 (2.67e+0) ≈	3.47e+2 (2.78e+0) −	3.47e+2 (4.79e+0) −	3.56e+2 (6.39e+0) +	3.50e+2 (2.64e+0) ≈	3.54e+2 (3.58e+0) +	3.55e+2 (3.36e+0) +	3.50e+2 (2.64e+0)
F24	4.22e+2 (2.95e+0) −	4.26e+2 (1.71e+0) ≈	4.26e+2 (2.42e+0) ≈	4.27e+2 (2.51e+0) +	4.25e+2 (1.56e+0) ≈	4.24e+2 (1.62e+0) −	4.23e+2 (3.74e+0) −	4.30e+2 (4.10e+0) +	4.26e+2 (1.58e+0) ≈	4.30e+2 (3.66e+0) +	4.27e+2 (2.77e+0) +	4.26e+2 (1.92e+0)
F25	3.87e+2 (7.80e−3) +	3.87e+2 (2.01e−2) +	3.87e+2 (2.00e−2) +	3.87e+2 (6.77e−3) +	3.87e+2 (1.62e−2) +	3.87e+2 (1.78e−2) +	3.87e+2 (1.13e−2) +	3.78e+2 (5.67e−2) −	3.87e+2 (2.47e−2) +	3.87e+2 (7.84e−3) +	3.87e+2 (6.32e−3) +	3.87e+2 (6.30e−3)
F26	8.73e+2 (9.17e+1) −	9.23e+2 (4.15e+1) +	9.37e+2 (5.34e+1) +	9.45e+2 (4.96e+1) +	9.29e+2 (3.66e+1) +	8.91e+2 (3.16e+1) ≈	8.36e+2 (7.01e+1) −	9.41e+2 (5.52e+1) +	9.25e+2 (3.96e+1) +	9.36e+2 (5.47e+1) +	9.28e+2 (5.44e+1) +	9.04e+2 (4.10e+1)
F27	4.99e+2 (7.41e+0) ≈	5.02e+2 (6.28e+0) ≈	4.78e+2 (5.25e+0) −	4.97e+2 (6.58e+0) −	5.02e+2 (6.40e+0) ≈	5.01e+2 (5.95e+0) ≈	4.99e+2 (8.79e+0) ≈	4.80e+2 (1.74e+1) −	5.03e+2 (5.00e+0) ≈	5.01e+2 (7.04e+0) ≈	5.04e+2 (4.80e+0) ≈	5.02e+2 (6.88e+0)
F28	3.15e+2 (3.86e+1) −	3.27e+2 (4.95e+1) ≈	3.25e+2 (4.83e+1) ≈	3.11e+2 (3.42e+1) ≈	3.21e+2 (4.32e+1) ≈	3.24e+2 (4.54e+1) −	3.19e+2 (4.12e+1) ≈	3.24e+2 (5.32e+1) −	3.21e+2 (4.36e+1) ≈	3.04e+2 (2.13e+1) −	3.15e+2 (3.86e+1) ≈	3.17e+2 (4.00e+1)
F29	4.31e+2 (1.26e+1) −	4.37e+2 (9.14e+0) ≈	3.95e+2 (2.80e+1) −	4.43e+2 (1.00e+1) +	4.33e+2 (9.77e+0) −	4.10e+2 (6.50e+0) −	4.28e+2 (1.05e+1) −	3.80e+2 (2.20e+1) −	4.35e+2 (7.77e+0) ≈	4.42e+2 (1.40e+1) +	4.33e+2 (7.84e+0) −	4.37e+2 (7.42e+0)
F30	2.00e+3 (3.05e+1) +	1.97e+3 (3.76e+1) −	8.25e+2 (3.52e+2) −	1.97e+3 (1.04e+1) ≈	1.98e+3 (3.94e+1) ≈	1.98e+3 (3.67e+1) ≈	1.98e+3 (2.14e+1) +	6.65e+2 (4.14e+2) −	1.98e+3 (3.84e+1) ≈	1.97e+3 (1.83e+1) ≈	1.99e+3 (4.51e+1) +	1.97e+3 (1.22e+1)
W	**11**	**10**	**4**	**10**	**9**	**7**	**12**	**7**	**10**	**11**	**15**	
L	**11**	**5**	**14**	**4**	**6**	**13**	**7**	**12**	**3**	**6**	**1**	
T	**7**	**14**	**11**	**15**	**14**	**9**	**10**	**10**	**16**	**12**	**13**	

Table 9
Performance comparison of ADCCDE with state-of-the-art DEs on 50–D functions.

Problem	ADE-DMRM	SCSS-L-SHADE	FODE	LSHADE-RSP	EaDE	DTDE	DE-EXP	MSDE	dFDB-LSHADE	mLSHADE-SPACMA	NSM-LSHADE-CnEpSin	ADCCDE
F1	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0)
F3	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0)
F4	6.34e+1 (4.74e+1) +	6.72e+1 (4.95e+1) +	5.05e+1 (4.21e+1) ≈	5.40e+1 (4.32e+1) +	8.96e+1 (5.11e+1) +	8.63e+1 (5.15e+1) +	7.28e+1 (4.96e+1) +	6.03e+1 (4.72e+1) ≈	7.36e+1 (4.81e+1) +	4.87e+1 (3.91e+1) +	4.87e+1 (3.91e+1) +	5.14e+1 (4.44e+1)
F5	1.51e+1 (1.94e+0) +	1.16e+1 (1.89e+0) −	1.43e+1 (2.88e+0) ≈	1.32e+1 (3.27e+0) ≈	8.02e+0 (3.45e+0) +	1.91e+0 (1.53e+0) −	1.86e+1 (5.76e+0) +	1.62e+1 (6.34e+0) ≈	1.21e+1 (2.42e+0) −	7.22e+0 (2.17e+0) −	7.22e+0 (2.17e+0) +	1.41e+1 (2.17e+0)
F6	2.19e−4 (2.86e−4) +	5.81e−8 (1.17e−7) ≈	3.92e−7 (7.01e−7) +	8.13e−8 (1.18e−7) ≈	2.10e−8 (6.20e−8) −	3.17e−8 (7.37e−8) ≈	3.66e−7 (4.68e−7) +	2.45e−8 (7.59e−8) −	2.02e−5 (1.43e−4) +	2.36e−10 (1.69e−9) −	2.36e−10 (1.69e−9) +	5.77e−8 (1.09e−7)
F7	6.36e+1 (1.79e+0) −	6.37e+1 (2.45e+0) −	6.58e+1 (3.60e+0) ≈	6.66e+1 (2.86e+0) +	6.14e+1 (2.57e+0) +	5.65e+1 (1.08e+0) −	7.00e+1 (5.07e+0) +	6.46e+1 (4.21e+0) ≈	6.31e+1 (1.78e+0) −	5.70e+1 (9.73e−1) −	5.70e+1 (9.73e−1) +	6.47e+1 (2.19e+0)
F8	1.57e+1 (2.35e+0) +	1.15e+1 (2.37e+0) −	1.43e+1 (3.05e+0) ≈	1.24e+1 (3.73e+0) −	8.10e+0 (3.06e+0) +	1.87e+0 (1.60e+0) −	1.81e+1 (4.51e+0) +	1.46e+1 (5.32e+0) ≈	1.22e+1 (2.32e+0) −	7.28e+0 (2.47e+0) −	7.28e+0 (2.47e+0) +	1.34e+1 (1.81e+0)
F9	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0)
F10	3.02e+3 (2.68e+2) ≈	3.20e+3 (2.80e+2) ≈	3.01e+3 (4.15e+2) ≈	3.49e+3 (3.90e+2) +	2.91e+3 (4.93e+2) ≈	1.59e+2 (9.38e+1) −	2.96e+3 (2.97e+2) −	2.21e+3 (1.11e+3) −	3.16e+3 (2.71e+2) ≈	3.61e+3 (6.40e+2) +	3.61e+3 (6.40e+2) ≈	3.11e+3 (3.05e+2)
F11	3.23e+1 (6.09e+0) +	3.30e+1 (4.61e+0) +	2.07e+1 (4.07e+0) ≈	2.50e+1 (3.58e+0) +	3.14e+1 (4.37e+0) +	3.26e+1 (3.89e+0) +	3.13e+1 (4.47e+0) +	2.06e+1 (3.81e+0) ≈	4.67e+1 (8.22e+0) +	2.99e+1 (4.56e+0) +	2.99e+1 (4.56e+0) +	2.05e+1 (2.13e+0)
F12	1.28e+3 (3.31e+2) +	2.18e+3 (5.08e+2) +	1.75e+3 (4.98e+2) +	1.63e+3 (4.55e+2) +	1.91e+3 (5.19e+2) +	2.02e+3 (5.41e+2) +	1.86e+3 (4.10e+2) +	1.93e+3 (4.60e+2) +	2.14e+3 (5.41e+2) +	1.13e+3 (3.42e+2) +	1.13e+3 (3.42e+2) +	8.68e+2 (3.92e+2)
F13	7.48e+1 (4.06e+1) +	4.04e+1 (2.79e+1) +	3.87e+1 (2.71e+1) ≈	3.33e+1 (2.08e+1) ≈	5.49e+1 (2.85e+1) +	4.47e+1 (2.78e+1) +	7.95e+1 (3.98e+1) +	3.98e+1 (2.32e+1) ≈	5.78e+1 (2.81e+1) +	2.44e+1 (1.70e+1) −	2.44e+1 (1.70e+1) +	3.48e+1 (2.81e+1)
F14	3.11e+1 (3.33e+0) +	2.54e+1 (1.91e+0) +	2.44e+1 (2.31e+0) +	2.30e+1 (1.61e+0) ≈	2.60e+1 (2.93e+0) +	2.47e+1 (4.69e+0) +	2.43e+1 (1.19e+0) +	2.50e+1 (2.40e+0) +	2.84e+1 (2.70e+0) +	2.41e+1 (2.05e+0) +	2.41e+1 (2.05e+0) +	2.27e+1 (1.15e+0)
F15	3.73e+1 (1.27e+1) +	2.68e+1 (3.55e+0) +	2.11e+1 (2.91e+0) ≈	2.09e+1 (2.07e+0) ≈	2.69e+1 (4.31e+0) +	2.75e+1 (3.54e+0) +	2.66e+1 (4.39e+0) +	2.22e+1 (3.15e+0) +	3.95e+1 (9.15e+0) +	2.16e+1 (1.91e+0) +	2.16e+1 (1.91e+0) +	2.02e+1 (1.53e+0)
F16	2.58e+2 (8.98e+1) −	3.47e+2 (1.26e+2) ≈	2.30e+2 (1.11e+2) −	3.24e+2 (1.42e+2) ≈	3.28e+2 (1.37e+2) ≈	1.33e+2 (2.47e+1) −	3.04e+2 (1.37e+2) ≈	2.10e+2 (8.82e+1) −	3.76e+2 (1.53e+2) +	4.15e+2 (1.95e+2) +	4.15e+2 (1.95e+2) ≈	3.16e+2 (1.24e+2)
F17	2.72e+2 (6.03e+1) +	2.25e+2 (7.69e+1) +	1.94e+2 (7.25e+1) ≈	2.12e+2 (1.00e+2) ≈	2.13e+2 (1.03e+2) ≈	5.83e+1 (5.48e+1) −	3.04e+2 (9.66e+1) +	1.40e+2 (6.75e+1) −	2.51e+2 (7.76e+1) +	2.37e+2 (9.46e+1) ≈	2.37e+2 (9.46e+1) +	1.95e+2 (7.66e+1)
F18	2.63e+1 (2.59e+0) +	2.65e+1 (2.72e+0) +	2.44e+1 (1.94e+0) +	2.31e+1 (1.40e+0) +	2.73e+1 (2.86e+0) +	3.29e+1 (6.22e+0) +	2.59e+1 (2.88e+0) +	2.49e+1 (1.91e+0) +	3.88e+1 (1.18e+1) +	2.40e+1 (1.99e+0) +	2.40e+1 (1.99e+0) +	2.23e+1 (1.26e+0)
F19	1.90e+1 (3.16e+0) +	1.61e+1 (3.00e+0) +	1.27e+1 (2.81e+0) ≈	1.13e+1 (2.62e+0) −	1.61e+1 (2.57e+0) +	1.04e+1 (2.16e+0) −	1.17e+1 (2.82e+0) −	1.19e+1 (2.29e+0) −	2.18e+1 (4.84e+0) +	1.34e+1 (2.61e+0) ≈	1.34e+1 (2.61e+0) +	1.33e+1 (2.55e+0)
F20	2.00e+2 (5.63e+1) +	1.50e+2 (5.58e+1) +	6.91e+1 (1.37e+1) −	1.24e+2 (6.37e+1) ≈	1.98e+2 (1.18e+2) ≈	2.34e+1 (2.51e+0) −	1.76e+2 (1.12e+2) +	3.55e+1 (9.42e+0) −	1.71e+2 (6.52e+1) +	1.56e+2 (9.20e+1) ≈	1.56e+2 (9.20e+1) +	1.19e+2 (4.22e+1)
F21	2.17e+2 (2.42e+0) +	2.12e+2 (2.60e+0) −	2.15e+2 (2.82e+0) ≈	2.13e+2 (3.82e+0) −	2.09e+2 (4.06e+0) ≈	2.03e+2 (1.74e+0) −	2.19e+2 (4.75e+0) +	2.17e+2 (6.40e+0) ≈	2.13e+2 (2.15e+0) −	2.17e+2 (1.22e+1) ≈	2.17e+2 (1.22e+1) +	2.15e+2 (2.40e+0)
F22	1.00e+2 (1.44e−14) −	3.09e+3 (1.36e+3) ≈	1.01e+3 (1.55e+3) −	2.34e+3 (1.92e+3) ≈	3.09e+3 (1.30e+3) +	3.39e+2 (1.50e+2) −	3.87e+2 (9.23e+2) −	8.05e+2 (8.90e+2) −	2.37e+3 (1.75e+3) ≈	2.23e+3 (1.93e+3) ≈	2.23e+3 (1.93e+3) −	2.85e+3 (1.49e+3)
F23	4.23e+2 (7.18e+0) −	4.29e+2 (4.15e+0) ≈	4.33e+2 (7.00e+0) +	4.32e+2 (6.18e+0) +	4.28e+2 (3.75e+0) −	4.22e+2 (7.16e+0) −	4.25e+2 (9.38e+0) −	4.35e+2 (1.18e+1) +	4.30e+2 (4.61e+0) +	4.44e+2 (7.81e+0) +	4.44e+2 (7.81e+0) +	4.28e+2 (4.72e+0)
F24	5.04e+2 (6.35e+0) −	5.05e+2 (2.61e+0) −	5.10e+2 (3.50e+0) +	5.09e+2 (4.01e+0) +	5.06e+2 (4.25e+0) −	5.00e+2 (2.73e+0) −	5.00e+2 (4.95e+0) −	5.11e+2 (4.88e+0) +	5.07e+2 (2.66e+0) ≈	5.15e+2 (8.88e+0) +	5.15e+2 (8.88e+0) +	5.08e+2 (3.06e+0)
F25	4.82e+2 (9.55e+0) +	4.81e+2 (2.88e+0) +	4.80e+2 (1.10e+1) −	4.81e+2 (2.32e+0) +	4.83e+2 (1.18e+1) +	4.81e+2 (3.48e+0) +	4.82e+2 (9.47e+0) +	4.73e+2 (1.45e+1) −	4.82e+2 (5.71e+0) +	4.81e+2 (3.15e+0) +	4.81e+2 (3.15e+0) +	4.81e+2 (2.62e+0)
F26	1.06e+3 (7.33e+1) −	1.14e+3 (5.07e+1) +	1.14e+3 (5.64e+1) +	1.12e+3 (5.17e+1) +	1.13e+3 (4.75e+1) ≈	1.04e+3 (4.79e+1) −	1.02e+3 (8.94e+1) −	1.16e+3 (5.33e+1) +	1.13e+3 (4.74e+1) +	1.14e+3 (5.12e+1) +	1.14e+3 (5.12e+1) +	1.09e+3 (5.51e+1)
F27	5.12e+2 (9.23e+0) −	5.27e+2 (1.20e+1) +	5.20e+2 (2.29e+1) ≈	5.12e+2 (1.27e+1) −	5.30e+2 (1.94e+1) +	5.29e+2 (1.38e+1) +	5.24e+2 (9.12e+0) +	4.98e+2 (8.19e+0) −	5.30e+2 (1.22e+1) +	5.17e+2 (1.00e+1) −	5.17e+2 (1.00e+1) ≈	5.23e+2 (1.78e+1)
F28	4.68e+2 (1.96e+1) +	4.62e+2 (1.16e+1) +	4.57e+2 (1.52e+1) −	4.59e+2 (1.81e−13) +	4.64e+2 (1.47e+1) ≈	4.62e+2 (1.16e+1) +	4.69e+2 (1.99e+1) +	4.76e+2 (2.51e+1) ≈	4.72e+2 (2.18e+1) +	4.59e+2 (1.70e−13) +	4.59e+2 (1.70e−13) +	4.60e+2 (9.47e+0)
F29	3.56e+2 (1.34e+1) −	3.62e+2 (1.23e+1) ≈	2.79e+2 (2.11e+1) −	3.67e+2 (1.52e+1) ≈	3.61e+2 (1.33e+1) ≈	3.05e+2 (8.42e+0) −	3.48e+2 (1.37e+1) −	2.55e+2 (1.06e+1) −	3.56e+2 (1.17e+1) −	3.71e+2 (1.92e+1) +	3.71e+2 (1.92e+1) −	3.62e+2 (1.08e+1)
F30	6.08e+5 (3.39e+4) −	6.53e+5 (7.54e+4) ≈	1.86e+3 (1.83e+3) −	6.11e+5 (4.34e+4) ≈	6.53e+5 (6.07e+4) +	6.50e+5 (7.12e+4) ≈	6.01e+5 (2.93e+4) −	1.19e+4 (2.17e+4) −	6.59e+5 (7.38e+4) ≈	6.12e+5 (3.83e+4) ≈	6.12e+5 (3.83e+4) ≈	6.42e+5 (6.51e+4)
W	**16**	**14**	**7**	**11**	**15**	**10**	**17**	**7**	**17**	**14**	**20**	
L	**9**	**5**	**7**	**4**	**3**	**14**	**8**	**11**	**5**	**6**	**2**	
T	**4**	**10**	**15**	**14**	**11**	**5**	**4**	**11**	**7**	**9**	**7**	

Table 10
Performance comparison of ADCCDE with state-of-the-art DEs on 100–D functions.

Problem	ADE-DMRM	SCSS-L-SHADE	FODE	LSHADE-RSP	EaDE	DTDE	DE-EXP	MSDE	dFDB-LSHADE	mLSHADE-SPACMA	NSM-LSHADE-CnEpSin	ADCCDE
F1	5.43e−6 (1.00e−5) +	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	4.12e−4 (6.14e−4) +	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0)
F3	2.63e−6 (4.88e−6) +	1.41e−4 (2.40e−4) +	2.79e−6 (3.37e−6) +	2.29e−7 (2.93e−7) +	1.10e−4 (1.86e−4) +	1.10e−4 (1.19e−4) +	4.34e−6 (5.07e−6) +	3.95e−6 (6.70e−6) +	1.37e−6 (1.19e−6) +	0.00e+0 (0.00e+0) −	0.00e+0 (0.00e+0) −	8.12e−8 (8.24e−8)
F4	1.86e+2 (2.56e+1) −	1.97e+2 (7.07e+0) ≈	1.74e+2 (4.04e+1) ≈	2.01e+2 (1.01e+1) +	1.98e+2 (9.22e+0) ≈	1.98e+2 (6.91e+0) ≈	1.82e+2 (3.90e+1) ≈	1.86e+2 (3.20e+1) −	1.90e+2 (2.66e+1) ≈	2.00e+2 (1.17e+1) +	2.05e+2 (9.90e+0) +	1.99e+2 (1.44e+1)
F5	3.99e+1 (3.88e+0) +	2.94e+1 (4.93e+0) +	3.69e+1 (1.18e+1) +	3.01e+1 (7.99e+0) +	2.43e+1 (6.52e+0) −	4.97e+0 (2.76e+0) −	3.57e+1 (5.08e+0) +	3.10e+1 (6.37e+0) +	3.72e+1 (5.57e+0) +	1.41e+1 (3.34e+0) −	5.99e+1 (7.01e+0) +	2.76e+1 (3.35e+0)
F6	8.14e−2 (2.91e−2) +	2.00e−3 (1.60e−3) +	1.00e−3 (1.60e−3) +	2.61e−5 (7.50e−5) +	2.10e−3 (1.90e−3) +	2.07e−3 (1.86e−3) +	5.30e−3 (5.00e−3) +	5.15e−4 (1.20e−3) +	4.40e−3 (3.14e−3) +	0.00e+0 (0.00e+0) −	4.59e−5 (1.58e−5) +	1.79e−6 (1.27e−6)
F7	1.32e+2 (2.99e+0) +	1.31e+2 (3.58e+0) +	1.59e+2 (1.08e+1) +	1.39e+2 (8.26e+0) +	1.30e+2 (3.96e+0) +	1.12e+2 (1.46e+0) −	1.38e+2 (5.43e+0) +	1.29e+2 (7.79e+0) ≈	1.40e+2 (4.03e+0) +	1.13e+2 (1.65e+0) −	1.69e+2 (6.43e+0) +	1.27e+2 (3.22e+0)
F8	3.92e+1 (4.20e+0) +	2.71e+1 (4.04e+0) −	3.63e+1 (9.37e+0) +	2.97e+1 (7.87e+0) ≈	2.30e+1 (6.03e+0) −	4.43e+0 (2.32e+0) −	3.66e+1 (4.70e+0) +	2.89e+1 (6.14e+0) ≈	3.86e+1 (4.34e+0) +	1.32e+1 (2.90e+0) −	5.89e+1 (6.33e+0) +	2.92e+1 (4.88e+0)
F9	1.57e+0 (8.89e−1) +	1.15e−1 (2.12e−1) +	0.00e+0 (0.00e+0) ≈	1.76e−3 (1.25e−2) ≈	1.17e−1 (1.75e−1) +	8.84e−2 (1.58e−1) +	1.26e+0 (7.86e−1) +	0.00e+0 (0.00e+0) ≈	3.17e−1 (4.35e−1) +	0.00e+0 (0.00e+0) ≈	0.00e+0 (0.00e+0 ≈	0.00e+0 (0.00e+0)
F10	9.25e+3 (5.14e+2) +	9.89e+3 (5.90e+2) +	1.04e+4 (8.07e+2) +	1.07e+4 (8.06e+2) +	9.36e+3 (8.86e+2) +	3.91e+2 (3.18e+2) −	8.94e+3 (5.38e+2) ≈	6.29e+3 (1.41e+3) −	1.03e+4 (5.79e+2) +	9.99e+3 (9.88e+2) +	1.03e+4 (5.27e+2) +	8.97e+3 (4.65e+2)
F11	1.29e+2 (3.59e+1) +	1.73e+2 (5.04e+1) +	1.08e+2 (3.13e+1) +	7.41e+1 (2.38e+1) +	1.70e+2 (4.79e+1) +	1.76e+2 (6.51e+1) +	2.50e+2 (6.60e+1) +	1.44e+2 (4.16e+1) +	4.05e+2 (9.00e+1) +	3.28e+1 (1.58e+1) ≈	4.84e+1 (3.50e+1) ≈	4.59e+1 (3.54e+1)
F12	5.77e+3 (1.35e+3) +	1.71e+4 (6.42e+3) +	1.80e+4 (6.29e+3) +	1.12e+4 (4.77e+3) +	1.64e+4 (6.63e+3) +	1.74e+4 (7.59e+3) +	2.13e+4 (1.45e+4) +	1.70e+4 (7.07e+3) +	2.43e+4 (1.06e+4) +	4.30e+3 (6.59e+2) −	4.56e+3 (6.72e+2) ≈	4.92e+3 (1.03e+3)
F13	4.19e+2 (4.13e+2) +	1.65e+2 (4.67e+1) +	1.20e+2 (3.19e+1) +	1.30e+2 (3.41e+1) +	1.68e+2 (4.97e+1) +	1.73e+2 (5.02e+1) +	8.06e+2 (2.68e+2) +	1.31e+2 (3.74e+1) +	4.80e+2 (4.05e+2) +	8.09e+1 (2.95e+1) +	1.06e+2 (3.56e+1) +	4.65e+1 (2.37e+1)
F14	7.08e+1 (1.49e+1) +	8.91e+1 (1.68e+1) +	6.21e+1 (1.00e+1) +	4.36e+1 (5.56e+0) +	8.76e+1 (1.70e+1) +	8.26e+1 (1.95e+1) +	1.00e+2 (2.12e+1) +	6.78e+1 (1.28e+1) +	2.50e+2 (2.86e+1) +	3.74e+1 (3.77e+0) ≈	5.13e+1 (8.50e+0) +	3.75e+1 (4.04e+0)
F15	1.75e+2 (4.40e+1) +	2.58e+2 (5.47e+1) +	1.97e+2 (5.12e+1) +	1.22e+2 (3.62e+1) +	2.57e+2 (5.13e+1) +	2.57e+2 (5.49e+1) +	2.52e+2 (5.24e+1) +	2.00e+2 (4.30e+1) +	2.56e+2 (4.86e+1) +	6.58e+1 (3.18e+1) +	8.01e+1 (2.87e+1) +	5.39e+1 (3.14e+1)
F16	1.37e+3 (2.97e+2) ≈	1.55e+3 (2.56e+2) +	1.18e+3 (3.18e+2) −	1.60e+3 (4.23e+2) +	1.45e+3 (2.84e+2) ≈	2.70e+2 (1.59e+2) −	1.07e+3 (2.92e+2) −	5.04e+2 (2.65e+2) −	1.76e+3 (2.22e+2) +	1.40e+3 (4.73e+2) ≈	1.30e+3 (2.47e+2) −	1.42e+3 (2.56e+2)
F17	1.10e+3 (2.29e+2) ≈	1.04e+3 (2.05e+2) ≈	9.25e+2 (2.20e+2) −	1.10e+3 (2.59e+2) ≈	1.01e+3 (2.94e+2) ≈	5.89e+1 (6.08e+1) −	1.04e+3 (2.22e+2) ≈	3.48e+2 (1.63e+2) −	1.17e+3 (1.93e+2) +	1.01e+3 (3.26e+2) ≈	1.06e+3 (1.76e+2) ≈	1.07e+3 (1.66e+2)
F18	9.16e+1 (2.08e+1) +	2.19e+2 (4.71e+1) +	2.11e+2 (4.44e+1) +	1.51e+2 (3.47e+1) +	2.12e+2 (4.22e+1) +	2.08e+2 (4.51e+1) +	2.06e+2 (4.09e+1) +	2.22e+2 (4.85e+1) +	2.35e+2 (4.37e+1) +	6.50e+1 (1.32e+1) +	6.67e+1 (1.70e+1) +	4.58e+1 (1.04e+1)
F19	9.25e+1 (1.56e+1) +	1.67e+2 (2.55e+1) +	1.32e+2 (2.38e+1) +	6.25e+1 (8.66e+0) +	1.67e+2 (2.60e+1) +	1.60e+2 (2.16e+1) +	1.89e+2 (3.52e+1) +	1.30e+2 (2.71e+1) +	1.76e+2 (2.00e+1) +	4.16e+1 (4.15e+0) −	5.28e+1 (6.64e+0) +	4.51e+1 (4.97e+0)
F20	1.71e+3 (1.62e+2) +	1.49e+3 (2.37e+2) +	1.07e+3 (1.73e+2) ≈	1.35e+3 (2.43e+2) +	1.44e+3 (2.74e+2) +	1.61e+2 (2.44e+1) −	1.44e+3 (2.31e+2) +	3.48e+2 (1.24e+2) −	1.51e+3 (1.85e+2) +	1.39e+3 (3.34e+2) +	1.25e+3 (1.86e+2) +	1.15e+3 (1.82e+2)
F21	2.65e+2 (5.98e+0) +	2.53e+2 (3.58e+0) ≈	2.57e+2 (8.47e+0) +	2.50e+2 (6.66e+0) −	2.50e+2 (6.52e+0) −	2.26e+2 (3.55e+0) −	2.60e+2 (5.84e+0) +	2.55e+2 (7.27e+0) ≈	2.58e+2 (6.43e+0) +	2.44e+2 (3.30e+0) −	2.85e+2 (1.30e+1) +	2.52e+2 (4.27e+0)
F22	9.74e+3 (2.01e+3) +	1.08e+4 (5.46e+2) +	1.07e+4 (9.77e+2) +	1.13e+4 (6.45e+2) +	1.01e+4 (8.79e+2) +	1.08e+3 (2.74e+2) −	9.81e+3 (1.43e+3) ≈	5.54e+3 (1.20e+3) −	1.13e+4 (5.18e+2) +	9.94e+3 (8.15e+2) ≈	1.08e+4 (5.79e+2) +	9.79e+3 (5.04e+2)
F23	5.83e+2 (1.69e+1) +	5.60e+2 (1.21e+1) ≈	5.61e+2 (8.68e+0) +	5.62e+2 (9.08e+0) +	5.62e+2 (1.02e+1) +	5.47e+2 (9.95e+0) −	5.66e+2 (1.33e+1) +	5.61e+2 (9.06e+0) +	5.68e+2 (1.01e+1) +	5.88e+2 (7.13e+0) +	6.07e+2 (1.83e+1) +	5.56e+2 (1.13e+1)
F24	8.89e+2 (1.51e+1) −	9.04e+2 (8.63e+0) +	9.07e+2 (7.72e+0) +	8.99e+2 (6.50e+0) ≈	9.05e+2 (7.32e+0) +	8.95e+2 (7.49e+0) −	8.89e+2 (1.16e+1) −	9.09e+2 (9.90e+0) +	9.13e+2 (7.44e+0) +	9.11e+2 (1.63e+1) +	9.16e+2 (9.11e+0) +	8.99e+2 (5.69e+0)
F25	7.04e+2 (4.10e+1) +	7.32e+2 (4.15e+1) +	7.25e+2 (3.52e+1) +	7.22e+2 (4.25e+1) +	7.40e+2 (3.16e+1) +	7.29e+2 (4.01e+1) +	7.14e+2 (4.06e+1) +	7.25e+2 (3.38e+1) +	7.47e+2 (2.89e+1) +	6.67e+2 (4.11e+1) ≈	6.90e+2 (4.41e+1) ≈	6.86e+2 (4.82e+1)
F26	3.04e+3 (8.98e+1) −	3.19e+3 (8.20e+1) +	3.25e+3 (9.30e+1) +	3.16e+3 (8.56e+1) +	3.23e+3 (9.17e+1) +	3.18e+3 (6.82e+1) +	3.06e+3 (1.22e+2) −	3.28e+3 (9.80e+1) +	3.31e+3 (8.68e+1) +	3.14e+3 (7.20e+1) ≈	3.04e+3 (1.10e+2) −	3.12e+3 (1.01e+2)
F27	5.70e+2 (1.40e+1) ≈	6.24e+2 (1.90e+1) +	5.00e+2 (4.98e−4) −	5.86e+2 (1.92e+1) +	6.20e+2 (1.76e+1) +	6.23e+2 (2.30e+1) +	6.16e+2 (1.52e+1) +	5.00e+2 (8.06e−4) −	6.26e+2 (1.70e+1) +	5.56e+2 (1.79e+1) −	5.72e+2 (1.75e+1) ≈	5.76e+2 (2.15e+1)
F28	5.27e+2 (3.06e+1) ≈	5.26e+2 (2.03e+1) +	4.98e+2 (5.89e+0) −	5.23e+2 (2.55e+1) ≈	5.32e+2 (2.56e+1) +	5.34e+2 (2.57e+1) +	5.32e+2 (3.85e+1) ≈	5.00e+2 (8.76e−4) −	5.25e+2 (2.67e+1) ≈	5.09e+2 (1.76e+1) ≈	5.13e+2 (1.69e+1) ≈	5.19e+2 (2.47e+1)
F29	1.10e+3 (1.66e+2) −	1.27e+3 (1.84e+2) ≈	8.90e+2 (1.78e+2) −	1.21e+3 (2.10e+2) ≈	1.24e+3 (1.91e+2) ≈	8.36e+2 (1.27e+2) −	1.12e+3 (1.41e+2) −	7.57e+2 (1.70e+2) −	1.23e+3 (1.55e+2) ≈	1.44e+3 (3.18e+2) +	1.16e+3 (1.37e+2) ≈	1.21e+3 (1.67e+2)
F30	2.38e+3 (1.68e+2) +	2.35e+3 (1.19e+2) +	1.03e+3 (1.39e+2) −	2.37e+3 (1.75e+2) +	2.34e+3 (1.22e+2) +	2.36e+3 (1.46e+2) +	2.68e+3 (2.13e+2) +	7.25e+2 (3.19e+2) −	2.40e+3 (1.35e+2) +	2.15e+3 (1.21e+2) −	2.18e+3 (7.05e+1) −	2.29e+3 (1.70e+2)
W	**21**	**22**	**19**	**21**	**21**	**15**	**20**	**14**	**25**	**9**	**16**	
L	**4**	**1**	**6**	**1**	**3**	**12**	**4**	**10**	**0**	**10**	**4**	
T	**4**	**6**	**4**	**7**	**5**	**2**	**5**	**5**	**4**	**10**	**9**	

Table 11
Comparison results with state-of-the-art DEs according to Wilcoxon's test.

Algorithm	30–D			50–D			100–D		
	win	tie	lose	win	tie	lose	win	tie	lose
LSHADE-RSP	**10**	15	4	**11**	14	4	**21**	7	1
FODE	**4**	11	14	**7**	15	7	**19**	4	6
SCSS-L-SHADE	**10**	14	5	**14**	10	5	**22**	6	1
EaDE	**9**	14	6	**15**	11	3	**21**	5	3
DE-EXP	**12**	10	7	**17**	4	8	**20**	5	4
DTDE	**7**	9	13	**10**	5	14	**15**	2	12
MSDE	**7**	10	12	**7**	11	11	**14**	5	10
ADE-DMRM	**11**	7	11	**16**	4	9	**21**	4	4
dFDB-LSHADE	**10**	16	3	**17**	7	5	**25**	4	0
mLSAHDE-SPACMA	**11**	12	6	**14**	9	6	**9**	10	10
NSM-LSHADE-CnEpSin	**15**	13	1	**20**	7	2	**16**	9	4

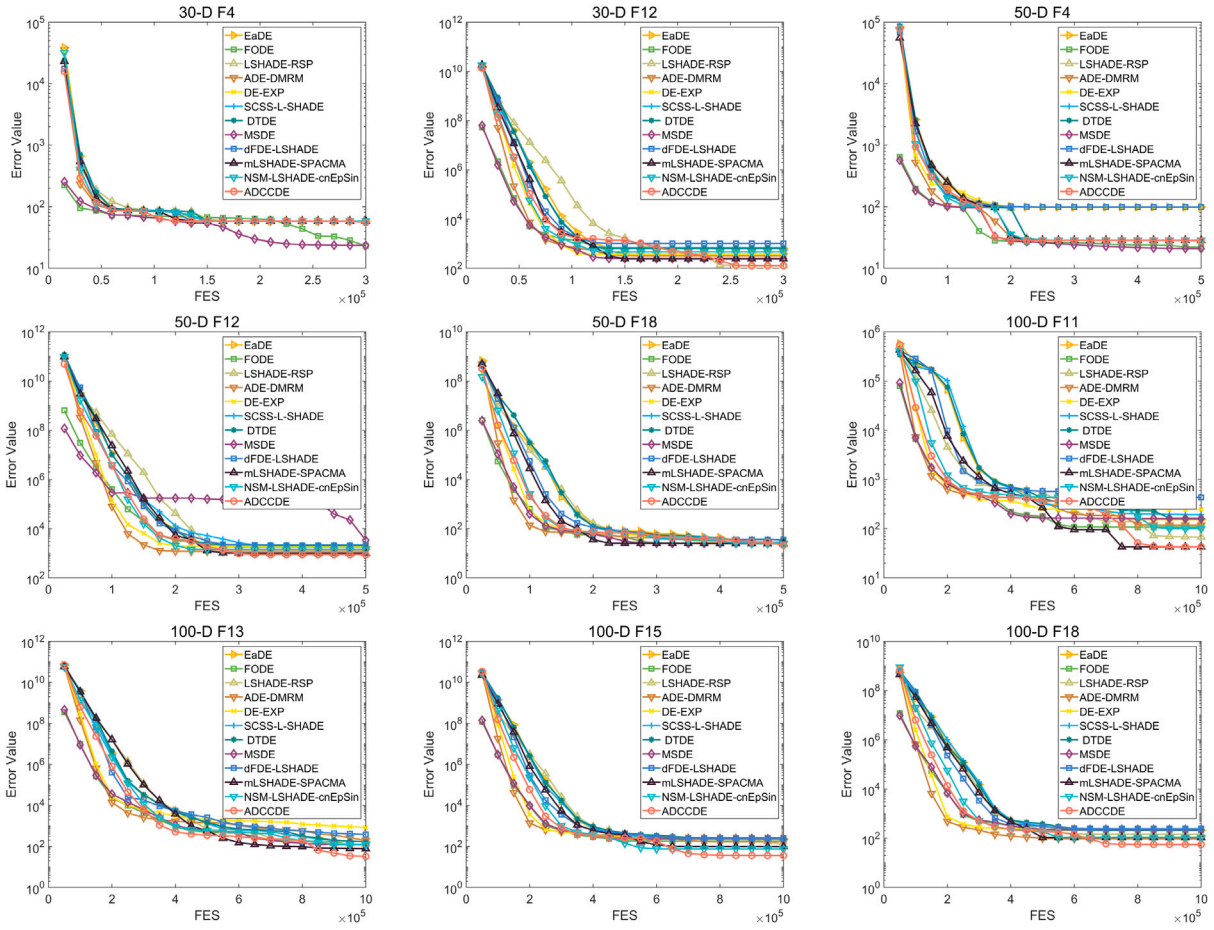


Fig. 4. Convergence plot of the compared DE algorithms on nine selected functions.

by two baseline algorithms, jSO and LSHADE-cnEpSin, through an evolutionary scale adaptation mechanism (ESA) (Zhang et al., 2024). Comparing ADCCDE with ESADE can clearly reveal the performance differences between ADCC and ESA. Their performance comparison on 30-D, 50-D, and 100-D functions is shown in Table 13.

From Table 13, it can be seen that: ADCCDE wins 7, 7, 8 in 30-D, 50-D, and 100-D respectively, and only underperforms ESADE on 50-D F5 and 100-D F5. Compared with ESADE, ADCCDE mainly has advantages in hybrid functions. As the dimensionality increases, the number of functions on which it outperforms ESADE increases from 3 to 4. When the dimensionality reaches 100, ADCCDE achieves performance improvements on 6 functions, while the performance on most of the remaining functions is tied with that of ESADE.

The cooperative subpopulations of ADCC, which integrate the advantages of two subpopulations, have a core mechanism of adaptive matching for exploration and exploitation at the individual dimension. This mechanism is highly consistent with the functional characteristics of randomly split sub-components of variables in hybrid functions. Although ESA dynamically selects an appropriate single strategy at the current stage, its optimization efficiency is reduced in hybrid functions due to cross-subcomponent interactions. This limitation explains why ADCCDE demonstrates superior performance on hybrid functions.

To further verify the advantages of ADCC over ESA, these two frameworks are integrated into the LSHADE-cnEpSin algorithm to construct the algorithms ESA-LSHADE-cnEpSin and ADCC-LSHADE-cnEpSin. Both algorithms adopt the "current-to-ci_*m*best" and "current-to-pbest" strategies. The former dynamically selects one of the two

Table 12
Overall performance ranking of the considered DEs.

Algorithm	Ranking
ADCCDE	**6.253**
FODE	6.351
ESADE	6.511
DTDE	6.557
MSDE	6.575
mLSHADE-SPACMA	6.851
LSHADE-RSP	7.684
ADE-DMRM	8.529
DE-EXP	8.730
EaDE	8.839
NSM-LSHADE-CnEpSin	8.925
LSHADE-cnEpSin	9.017
SCSS-L-SHADE	9.155
jSO	9.316
dFDB-LSHADE	10.707

strategies via a threshold to generate offspring, while the latter, on the basis of generating offspring using the two strategies, additionally incorporates an offspring fusion module that participates in the competition.

Performance comparisons of ADCC with ESA are presented in Table 14. On unimodal and simple multimodal functions, regardless of whether it is 30-D, 50-D or 100-D, ADCC-LSHADE-cnEpSin outperforms ESA-LSHADE-cnEpSin on F5, F7, and F8. Additionally, it has an advantage on 100-D F6. For hybrid functions, improvements are observed on 30-D F11, 50-D F13, 50-D F20, 100-D F15, and 100-D F16, while the rest are tied. For composition functions, the "W/L" ratio for both 30-D and 50-D is "3/1". When the dimensionality increases to 100, ADCC-LSHADE-cnEpSin becomes more competitive with the "W/L" ratio changing to "6/1".

Compared with being equipped with algorithm modules, being equipped with strategy modules yields remarkably better performance in high-dimensional problems. This may be attributed to the absence of conflicts in mechanisms and parameters among different algorithms, and such an advantage is particularly prominent in composition functions. Composition functions have multiple local optima, and all their basic functions are processed through rotation, which leads to strong correlations among variables. In ADCC, the fusion subpopulation adopts an allocation method based on the dispersion degree of dimensions. Actually, single-strategy subpopulations produce the same effect on dimensions with high correlation. Therefore, the allocation method based on dispersion degree can make the subpopulations with strong correlations consistent in the direction of exploitation or exploration in the next generation. That is to say, it can break the problems caused by rotation and effectively help escape from local optima. For ESA, which also adopts dynamic competition, the global homogenization caused by a single strategy at a certain stage may ignore the correlations among variables.

4.5. Applications in real-world optimization

To further assess the performance of ADCCDE against state-of-the-art DE algorithms in real-world optimization scenarios, eight practical application problems covering multiple fields are selected from the

Table 13
Performance comparisons of ADCCDE with ESADE on 30–D, 50–D and 100–D functions.

Problem	30–D					50–D					100–D				
	ESADE			ADCCDE		ESADE			ADCCDE		ESADE			ADCCDE	
	mean	std	sig	mean	std	mean	std	sig	mean	std	mean	std	sig	mean	std
F1	0.00e+00	0.00e+00	≈	0.00e+00	0.00e+00	0.00e+00	0.00e+00	≈	0.00e+00	0.00e+00	0.00e+00	0.00e+00	≈	0.00e+00	0.00e+00
F3	0.00e+00	0.00e+00	≈	0.00e+00	0.00e+00	0.00e+00	0.00e+00	≈	0.00e+00	0.00e+00	1.62e−07	1.81e−07	+	8.12e−08	8.24e−08
F4	5.03e+01	2.03e+00	+	4.90e+01	2.16e+00	7.45e+01	4.80e+01	+	5.14e+01	4.44e+01	1.98e+02	1.32e+01	≈	1.99e+02	1.44e+01
F5	7.56e+00	1.67e+00	≈	7.75e+00	1.47e+00	1.27e+01	2.09e+00	−	1.41e+01	2.17e+00	2.57e+01	4.02e+00	−	2.76e+01	3.35e+00
F6	6.98e−08	1.54e−07	+	0.00e+00	0.00e+00	7.76e−07	1.10e−06	+	5.77e−08	1.09e−07	1.90e−05	9.60e−06	+	1.79e−06	1.27e−06
F7	3.85e+01	1.69e+00	≈	3.89e+01	1.48e+00	6.45e+01	2.15e+00	≈	6.47e+01	2.19e+00	1.26e+02	2.79e+00	≈	1.27e+02	3.22e+00
F8	7.75e+00	1.52e+00	≈	8.24e+00	1.44e+00	1.29e+01	2.92e+00	≈	1.34e+01	1.81e+00	2.77e+01	3.43e+00	≈	2.92e+01	4.88e+00
F9	0.00e+00	0.00e+0	≈	0.00e+00	0.00e+00	0.00e+00	0.00e+00	≈	0.00e+00	0.00e+00	0.00e+00	0.00e+00	≈	0.00e+00	0.00e+00
F10	1.51e+03	2.27e+02	≈	1.47e+03	2.20e+02	3.10e+03	3.59e+02	≈	3.11e+03	3.05e+02	8.86e+03	5.19e+02	≈	8.97e+03	4.65e+02
F11	8.99e+00	1.67e+01	≈	8.94e+00	1.73e+01	2.18e+01	1.99e+00	+	2.05e+01	2.13e+00	5.60e+01	3.71e+01	+	4.59e+01	3.54e+01
F12	1.20e+02	1.02e+02	≈	1.35e+02	8.33e+01	8.41e+02	3.17e+02	≈	8.68e+02	3.92e+02	5.41e+03	1.20e+03	+	4.92e+03	1.03e+03
F13	1.76e+01	5.44e+00	≈	1.74e+01	3.99e+00	5.47e+01	3.50e+01	+	3.48e+01	2.81e+01	7.88e+01	2.96e+01	+	4.65e+01	2.37e+01
F14	2.01e+01	4.09e+00	≈	2.10e+01	2.70e+00	2.32e+01	1.24e+00	+	2.27e+01	1.15e+00	3.74e+01	4.05e+00	≈	3.75e+01	4.04e+00
F15	3.48e+00	2.15e+00	+	9.79e−01	6.64e−01	2.06e+01	1.91e+00	≈	2.02e+01	1.53e+00	7.52e+01	3.53e+01	+	5.39e+01	3.14e+01
F16	2.06e+01	2.58e+01	≈	3.14e+01	4.70e+01	3.20e+02	1.26e+02	≈	3.16e+02	1.24e+02	1.45e+03	2.17e+02	≈	1.42e+03	2.56e+02
F17	3.51e+01	4.99e+00	+	3.13e+01	6.21e+00	2.47e+02	7.90e+01	+	1.95e+02	7.66e+01	1.05e+03	1.83e+02	≈	1.07e+03	1.66e+02
F18	2.03e+01	2.70e+00	≈	2.06e0+1	1.99e−01	2.25e+01	1.24e+00	≈	2.23e+01	1.26e+00	6.01e+01	1.44e+01	+	4.58e+01	1.04e+01
F19	5.66e+00	1.33e+00	≈	5.47e+00	1.77e+00	1.28e+01	2.48e+00	≈	1.33e+01	2.55e+00	4.73e+01	4.50e+00	+	4.51e+01	4.97e+00
F20	3.37e+01	4.92e+00	+	2.90e+01	7.20e+00	1.19e+02	4.39e+01	≈	1.19e+02	4.22e+01	1.17e+03	1.83e+02	≈	1.15e+03	1.82e+02
F21	2.08e+02	1.58e+00	≈	2.08e+02	1.65e+00	2.15e+02	2.82e+00	≈	2.15e+02	2.40e+00	2.51e+2	4.64e+00	≈	2.52e+02	4.27e+00
F22	1.00e+02	6.39e−14	≈	1.00e+02	1.44e−14	2.59e+03	1.56e+03	≈	2.85e+03	1.49e+03	9.69e+03	4.80e+02	≈	9.79e+03	5.04e+02
F23	3.50e+02	2.51e+00	≈	3.50e+02	2.64e+00	4.28e+02	4.71e+00	≈	4.28e+02	4.72e+00	5.53e+02	7.55e+00	≈	5.56e+02	1.13e+01
F24	4.26e+02	1.99e+00	≈	4.26e+02	1.92e+00	5.07e+02	2.77e+00	≈	5.08e+02	3.06e+00	8.99e+02	5.53e+00	≈	8.99e+02	5.69e+00
F25	3.87e+02	5.30e−03	≈	3.87e+02	6.30e−03	4.80e+02	1.53e+00	≈	4.81e+02	2.62e+00	6.72e+02	5.00e+01	≈	6.86e+02	4.82e+01
F26	8.91e+02	3.47e+01	≈	9.04e+02	4.10e+01	1.08e+03	3.83e+01	≈	1.09e+03	5.51e+01	3.08e+03	9.16e+01	≈	3.12e+03	1.01e+02
F27	5.00e+02	6.31e+00	≈	5.02e+02	6.88e+00	5.20e+02	1.11e+01	≈	5.23e+02	1.78e+01	5.74e+02	1.81e+01	≈	5.76e+02	2.15e+01
F28	3.06e+02	2.45e+01	+	3.17e+02	4.00e+01	4.58e+02	1.83e−01	+	4.60e+02	9.47e+00	5.16e+02	2.61e+01	≈	5.19e+02	2.47e+01
F29	4.36e+02	1.30e+01	≈	4.37e+02	7.42e+00	3.58e+02	1.15e+01	≈	3.62e+02	1.08e+01	1.17e+03	1.52e+02	≈	1.21e+03	1.67e+02
F30	1.98e+03	4.07e+01	+	1.97e+03	1.22e0+1	6.48e+05	6.87e+04	≈	6.42e+05	6.51e+4	2.25e+03	1.09e+02	≈	2.29e+03	1.70e+02
W	**7**					**7**					**8**				
L	**0**					**1**					**1**				
T	**22**					**21**					**20**				

Table 14
Performance comparisons of ADCC–LSHADE–cnEpSin with ESA–LSHADE–cnEpSin on 30–D, 50–D and 100–D functions.

Problem	30–D					50–D					100–D				
	ESA–LSHADE–cnEpSin			ADCC–LSHADE–cnEpSin		ESA–LSHADE–cnEpSin			ADCC–LSHADE–cnEpSin		ESA–LSHADE–cnEpSin			ADCC–LSHADE–cnEpSin	
	mean	std	sig	mean	std	mean	std	sig	mean	std	mean	std	sig	mean	std
F1	0.00e+00	0.00e+00	≈	0.00e+00	0.00e+00	0.00e+00	0.00e+00	≈	0.00e+00	0.00e+00	0.00e+00	0.00e+00	≈	0.00e+00	0.00e+00
F3	0.00e+00	0.00e+00	≈	0.00e+00	0.00e+00	0.00e+00	0.00e+00	≈	0.00e+00	0.00e+00	0.00e+00	0.00e+00	≈	0.00e+00	0.00e+00
F4	4.29e+01	3.94e+00	≈	4.40e+01	3.18e+00	4.73e+01	4.06e+01	≈	5.24e+01	4.67e+01	2.01e+02	7.89e+00	≈	2.00e+02	8.43e+00
F5	1.05e+01	2.49e+00	+	9.45e+00	1.60e+00	2.01e+01	6.95e+00	+	1.66e+01	5.43e+00	4.72e+01	7.65e+00	+	3.43e+01	6.19e+00
F6	1.61e−08	8.05e−08	≈	3.07e−09	1.07e−08	7.88e−07	9.33e−07	≈	5.72e−07	6.12e−07	6.56e−05	2.51e−05	+	4.18e−05	1.80e−05
F7	4.13e+01	2.39e+00	+	3.92e+01	1.92e+00	7.04e+01	4.99e+00	+	6.63e+01	3.97e+00	1.49e+02	4.93e+00	+	1.42e+02	4.90e+00
F8	1.15e+01	2.13e+00	+	9.24e+00	1.93e+00	2.05e+01	6.36e+00	+	1.76e+01	4.83e+00	4.40e+01	1.03e+01	+	3.34e+01	5.98e+00
F9	0.00e+00	0.00e+00	≈	0.00e+00	0.00e+00	0.00e+00	0.00e+00	≈	0.00e+00	0.00e+00	0.00e+00	0.00e+00	≈	0.00e+00	0.00e+00
F10	1.45e+03	2.29e+02	≈	1.41e+03	2.09e+02	3.14e+03	2.91e+02	≈	3.11e+03	3.56e+02	1.01e+04	6.03e+02	≈	1.01e+04	5.91e+02
F11	1.84e+01	2.33e+01	+	1.08e+01	1.79e+01	2.19e+01	1.90e+00	≈	2.22e+01	2.61e+00	5.53e+01	3.16e+01	≈	4.67e+01	2.90e+01
F12	3.82e+02	2.37e+02	≈	3.69e+02	1.82e+2	1.22e+03	4.35e+02	≈	1.34e+03	2.95e+02	4.48e+03	6.14e+02	≈	4.53e+03	6.13e+02
F13	1.53e+01	6.17e+00	≈	1.52e+01	6.58e+00	7.59e+01	3.47e+01	+	4.73e+01	3.27e+01	1.16e+02	3.86e+01	≈	1.03e+02	2.88e+01
F14	2.11e+01	2.89e+00	≈	2.05e+01	4.04e+00	2.59e+01	1.78e+00	≈	2.63e+01	2.94e+00	5.11e+01	7.65e+00	≈	5.20e+01	9.75e+00
F15	3.65e+00	2.04e+00	≈	3.03e+00	1.82e+00	2.58e+01	3.78e+00	≈	2.43e+01	3.60e+00	9.25e+01	2.45e+01	+	8.14e+01	2.95e+01
F16	1.37e+01	3.48e+00	≈	1.70e+01	2.30e+01	2.73e+02	1.10e+02	≈	2.56e+02	1.14e+02	1.20e+03	2.44e+02	+	9.43e+02	2.84e+02
F17	2.80e+01	6.55e+00	≈	2.71e+01	5.84e+00	2.08e+02	6.49e+01	≈	2.12e+02	6.85e+01	8.83e+02	1.78e+02	≈	8.26e+02	1.54e+02
F18	2.08e+01	2.90e+00	≈	2.12e+01	8.07e−01	2.45e+01	2.35e+00	≈	2.43e+01	1.98e+00	7.47e+01	1.81e+01	≈	6.97e+01	1.53e+01
F19	5.88e+00	1.65e+00	≈	5.77e+00	1.67e+00	1.64e+1	2.55e+00	≈	1.69e+01	3.49e+00	5.64e+01	5.45e+00	≈	5.54e+01	5.51e+00
F20	3.08e+01	5.58e+00	≈	3.10e+01	7.87e+00	1.18e+02	3.50e+01	+	1.03e+02	1.54e+01	1.11e+03	1.71e+02	≈	1.05e+03	1.37e+02
F21	2.11e+02	2.76e+00	+	2.09e+02	2.46e+00	2.23e+02	6.83e+00	+	2.19e+02	5.86e+00	2.69e+02	1.01e+01	+	2.59e+02	5.93e+00
F22	1.00e+02	1.44e−14	≈	1.00e+02	6.39e−14	1.66e+03	1.70e+03	≈	1.24e+03	1.63e+03	1.03e+04	7.00e+02	+	9.83e+03	7.27e+02
F23	3.54e+02	2.89e+00	+	3.52e+02	3.83e+00	4.36e+02	5.66e+00	+	4.31e+02	5.72e+00	5.89e+02	8.39e+00	+	5.74e+02	9.16e+00
F24	4.27e+02	2.56e+00	≈	4.27e+02	2.19e+00	5.11e+02	5.60e+00	≈	5.10e+02	4.34e+00	9.12e+02	1.21e+01	+	9.00e+02	9.66e+00
F25	3.87e+02	7.00e−03	−	3.87e+02	7.00e−03	4.81e+02	3.12e+00	≈	4.81e+02	2.86e+00	6.70e+02	4.52e+01	−	6.92e+02	4.69e+01
F26	9.29e+02	4.34e+01	+	9.01e+02	4.33e+01	1.16e+03	8.68e+01	+	1.12e+03	7.69e+01	3.06e+03	8.92e+01	+	3.01e+03	6.39e+01
F27	5.04e+02	4.62e+00	≈	5.05e+02	5.94e+00	5.26e+02	1.49e+01	≈	5.29e+02	1.56e+01	5.91e+02	1.96e+01	≈	5.88e+02	1.56e+01
F28	3.13e+02	3.60e+01	≈	3.21e+02	4.36e+01	4.59e+02	1.17e+01	−	4.60e+02	1.17e+01	5.14e+02	1.60e+01	≈	5.12e+02	2.04e+01
F29	4.35e+02	6.69e+00	≈	4.34e+02	6.37e+00	3.52e+02	1.04e+01	≈	3.50e+02	8.78e+00	1.12e+03	1.49e+02	+	1.05e+03	1.23e+02
F30	1.99e+03	5.24e+01	≈	2.00e+03	7.21e+01	6.53e+05	8.34e+04	≈	6.47e+05	8.35e+04	2.35e+03	1.45e+02	≈	2.38e+03	1.96e+02
W	**7**					**8**					**12**				
L	**1**					**1**					**1**				
T	**21**					**20**					**16**				

CEC2011 (Das and Suganthan, 2010) for experiments. These include Lennard-Jones Potential Problem (P1), Spread Spectrum Radar Polly phase Code Design Problem (P2), Large Scale Transmission Pricing Problem (P3), Static Economic Load Dispatch Problem No. 11.5 and 11.6 with different sizes and constraint (P4 and P5), Hydrothermal Scheduling Problem No. 11.8 and 11.20 (P6 and P7), and Messenger: Spacecraft Trajectory Optimization Problem (P8). For each algorithm and each problem, every run is terminated after 150,000 function evaluations. For the eight benchmark problems, the mean, standard deviation, best, worst, and *p*-values of the tested algorithm are reported, along with the Wilcoxon rank-sum test results of ADCCDE, as summarized in Table 15.

As presented in Table 15, the ADCCDE algorithm achieves distinct advantages on problems of P4 and P5, outperforming 11 advanced DE variants in all cases with no defeats. On problems of P6 and P7, it fails to outperform the two algorithms, dFDB-LSHADE and NSM-LSHADE-CnEpSin, with the rest of the results being win or tie. Similarly, among the comparisons with numerous algorithms, only the performance on P1 and P8 is suboptimal, with the "W/L" ratios of these two problems being "5/2" and "9/2" respectively. On P1 problem, it ties with most of the compared algorithms. Although it is outperformed by DE-EXP, ADE-DMRM and dFDB-LSHADE on P3 problems, it wins in most of the other comparisons, with a "W/L" ratio of "10/3". On P2 problem, it also suffers three defeats, with a slightly lower "W/L" ratio of "7/3" compared with that on P3.

Table 16 summarizes the results of the Wilcoxon rank-sum test of the ADCCDE algorithm against 11 state-of-the-art DE algorithms on eight problems. Obviously, our ADCCDE algorithm achieves results of "win" or "tie" in most comparison scenarios. In the vertical comparison with these 11 advanced algorithms, ADCCDE suffers no defeats against six of them. Among which, it wins as many as 7 problems against both LSHADE-RSP and mLSHADE-SPACMA. However, against the three algorithms LSHADE-cnEpSin, dFDB-LSHADE and NSM-LSHADE-CnEpSin, it performs less satisfactorily, with the "W/L" ratios being "5/2", "4/3" and "3/2" respectively.

To conduct the significance analysis of the overall performance differences among multiple algorithms on multiple tests, the Friedman test is adopted to eliminate the interference of randomness and instability, the results of which are presented in Table 17. With NSM-LSHADE-CnEpSin as the midpoint, a cliff-like gap emerges between the average ranks of the algorithms behind it and those in front of it. ESADE, EaDE, DTDE and SCSS-L-SHADE have similar ranks behind it. Similarly, dFDB-LSHADE has a rank close to that of ADCCDE, yet ADCCDE still ranks first with an advantageous average rank of 3.750.

5. Conclusion

This paper proposes an adaptive dual-module competition and cooperation (ADCC) optimization framework, aiming to address the dynamic balance challenge between exploration and exploitation in complex optimization problems. The core of ADCC lies in designing two modules with complementary characteristics and their dynamic integration scheme to achieve adaptive adjustment of strategies. One module is oriented toward global exploration, whereas the other emphasizes local exploitation to intensify the search within high-quality regions. The fusion module integrates the dimensional fragments of the trial vectors generated by the original dual modules at the individual level based on the fitness of the offspring and the iteration progress,

Table 15
Performance comparisons with state-of-the-art DEs on real-world problems.

Problem	Algorithm															
		jSO	LSHADE-cnEpSin	LSHADE-RSP	FODE	SCSS-L-SHADE	EaDE	DTDE	DE-EXP	ADE-DMRM	MSDE	ESADE	dFDB-LSHADE	mLSAHDE-SPACMA	NSM-LSHADE-CnEpSin	ADCCDE
P1	Mean	−2.62e+01	−2.61e+01	−2.45e+01	−1.76e+01	−2.63e+01	−2.62e+01	−1.09e+01	−2.66e+01	−2.64e+01	−2.71e+01	−2.64e+01	−2.75e+01	−2.75e+01	−2.74e+01	−2.64e+01
	STD	7.96e−01	5.56e−01	1.41e+00	9.08e−01	4.90e−01	4.86e−01	2.01e+00	5.23e−01	4.86e−01	6.76e−01	6.18e−01	4.24e−01	3.99e−01	4.77e−01	5.19e−01
	Best	−2.45e+01	−2.50e+01	−2.01e+01	−1.95e+01	−2.53e+01	−2.52e+01	−7.03e+00	−2.56e+01	−2.56e+01	−2.54e+01	−2.53e+01	−2.84e+01	−2.83e+01	−2.83e+01	−2.54e+01
	Worst	−2.78e+01	−2.75e+01	−2.68e+01	−1.58e+01	−2.71e+01	−2.72e+01	−1.43e+01	−2.84e+01	−2.78e+01	−2.84e+01	−2.80e+01	−2.72e+01	−2.69e+01	−2.65e+01	−2.75e+01
	p-value	2.97e−01	5.01e−02	4.31e−08	3.02e−11	6.20e−01	1.96e−01	3.02e−11	1.05e−01	8.30e−01	8.15e−05	7.62e−01	3.78e−02	3.82e−09	4.36e−02	/
	+/−/≈	≈	≈	+	+	≈	≈	+	≈	≈	−	≈	+	−	+	/
P2	Mean	1.05e+00	5.00e−01	1.14e+00	1.29e+00	1.16e+00	1.07e+00	7.16e−01	1.10e+00	1.05e+00	6.17e−01	1.05e+00	1.14e+00	1.10e+00	1.02e+00	1.04e+00
	STD	1.02e−01	0.00e+00	1.43e−01	1.24e−01	9.00e−02	1.05e−01	1.46e−01	8.55e−02	1.03e−01	1.11e−01	8.75e−02	7.82e−02	1.16e−01	4.76e−02	9.34e−02
	Best	8.85e−01	5.00e−01	7.71e−01	1.01e+00	9.30e−01	6.83e−01	5.00e−01	8.99e−01	7.25e−01	5.00e−01	8.65e−01	1.00e+00	8.68e−01	8.93e−01	8.54e−01
	Worst	1.24e+00	5.00e−01	1.40e+00	1.47e+00	1.35e+00	1.22e+00	9.71e−01	1.24e+00	1.19e+00	8.88e−01	1.20e+00	1.30e+00	1.29e+00	1.10e+00	1.22e+00
	p-value	7.73e−01	1.21e−12	1.17e−03	1.43e−08	4.74e−06	1.09e−01	2.61e−10	1.08e−02	4.29e−01	3.68e−11	6.84e−01	1.17e−04	2.32e−02	4.84e−02	/
	+/−/≈	≈	−	+	+	+	≈	−	+	≈	−	≈	+	+	+	/
P3	Mean	2.71e+03	3.28e+03	2.30e+03	3.70e+04	2.31e+03	2.37e+03	2.44e+03	7.70e+02	1.55e+03	4.43e+03	2.65e+03	1.36e+03	2.55e+04	1.80e+03	1.76e+03
	STD	5.34e+02	1.12e+03	5.53e+02	5.93e+03	5.69e+02	5.28e+02	5.65e+02	2.14e+02	3.59e+02	4.79e+02	6.41e+02	1.17e+02	1.02e+00	3.57e+02	4.67e+02
	Best	1.41e+03	1.36e+03	1.35e+03	1.76e+04	9.57e+02	1.28e+03	1.09e+03	3.51e+02	8.93e+02	3.55e+03	1.53e+03	1.18e+03	2.55e+04	1.37e+03	9.62e+02
	Worst	3.84e+03	6.46e+03	3.67e+03	4.43e+04	3.88e+03	3.72e+03	3.51e+03	1.33e+03	2.51e+03	5.29e+03	3.87e+03	1.61e+03	2.55e+04	2.96e+03	2.81e+03
	p-value	5.53e−08	4.69e−08	2.53e−04	3.02e−11	1.32e−04	4.08e−05	1.75e−05	1.09e−10	4.21e−02	3.02e−11	1.11e−06	2.39e−04	3.02e−11	8.42e−01	/
	+/−/≈	+	+	+	+	+	+	+	−	−	+	+	−	+	≈	/
P4	Mean	3.27e+04	3.28e+04	3.27e+04	3.27e+04	3.27e+04	3.27e+04	3.27e+04	3.27e+04	3.28e+04	3.28e+04	3.27e+04	3.27e+04	1.00e+30	3.27e+04	3.27e+04
	STD	5.73e−01	4.58e+01	2.74e−01	2.56e+00	1.54e−01	1.51e−01	2.90e−01	1.25e+00	3.17e+01	1.56e+01	2.88e−01	2.16e−01	1.43e+14	1.08e+01	2.14e−01
	Best	3.27e+04	3.27e+04	3.27e+04	3.27e+04	3.27e+04	3.27e+04	3.27e+04	3.27e+04	3.27e+04	3.27e+04	3.27e+04	3.27e+04	1.00e+30	3.27e+04	3.27e+04
	Worst	3.27e+04	3.29e+04	3.27e+04	3.27e+04	3.27e+04	3.27e+04	3.27e+04	3.27e+04	3.29e+04	3.28e+04	3.27e+04	3.27e+04	1.00e+30	3.28e+04	3.27e+04
	p-value	2.05e−03	4.98e−11	1.17e−03	8.64e−01	7.17e−01	4.12e−01	2.05e−03	4.11e−07	7.12e−09	4.50e−11	9.52e−04	4.36e−02	1.21e−12	1.60e−03	/
	+/−/≈	+	+	+	≈	≈	≈	+	+	+	+	+	+	+	+	/
P5	Mean	1.24e+05	1.25e+05	1.24e+05	1.23e+05	1.24e+05	1.24e+05	1.24e+05	1.24e+05	1.24e+05	1.25e+05	1.23e+05	1.23e+05	7.24e+08	1.24e+05	1.23e+05
	STD	5.26e+02	5.30e+02	5.25e+02	4.65e+02	5.17e+02	5.02e+02	4.67e+02	7.28e+02	4.69e+02	6.43e+02	6.30e+02	4.67e+02	2.33e+07	4.14e+02	4.59e+02
	Best	1.23e+05	1.23e+05	1.23e+05	1.22e+05	1.23e+05	1.23e+05	1.23e+05	1.23e+05	1.23e+05	1.24e+05	1.22e+05	1.22e+05	6.81e+08	1.22e+05	1.22e+05
	Worst	1.25e+05	1.26e+05	1.25e+05	1.24e+05	1.25e+05	1.25e+05	1.25e+05	1.25e+05	1.25e+05	1.26e+05	1.25e+05	1.24e+05	7.55e+08	1.25e+05	1.24e+05
	p-value	1.17e−04	1.21e−10	6.28e−06	3.26e−01	5.09e−06	2.57e−07	1.73e−06	1.73e−07	2.67e−09	4.50e−11	1.70e−02	1.15e−01	3.01e−11	5.20e−01	/
	+/−/≈	+	+	+	≈	+	+	+	+	+	+	+	≈	+	≈	/
P6	Mean	9.33e+05	9.35e+05	9.33e+05	9.53e+05	9.33e+05	9.32e+05	9.33e+05	4.91e+06	4.18e+06	9.41e+05	9.34e+05	9.26e+05	2.57e+07	9.24e+05	9.32e+05
	STD	1.96e+03	2.02e+03	1.25e+03	3.59e+03	1.63e+03	1.25e+03	1.33e+03	9.19e+05	4.72e+05	1.29e+03	1.42e+03	6.17e+02	4.82e+02	5.05e+02	1.44e+03
	Best	9.30e+05	9.31e+05	9.30e+05	9.44e+05	9.30e+05	9.30e+05	9.30e+05	2.93e+06	3.33e+06	9.39e+05	9.32e+05	9.25e+05	2.57e+07	9.24e+05	9.29e+05
	Worst	9.38e+05	9.39e+05	9.35e+05	9.60e+05	9.39e+05	9.36e+05	9.34e+05	6.81e+06	5.62e+06	9.43e+05	9.37e+05	9.27e+05	2.57e+07	9.26e+05	9.35e+05
	p-value	1.77e−03	1.56e−08	3.15e−02	3.02e−11	1.96e−01	1.96e−01	1.91e−02	3.02e−11	3.02e−11	3.02e−11	5.19e−07	3.02e−11	3.02e−11	3.02e−11	/
	+/−/≈	+	+	+	+	≈	≈	+	+	+	+	+	−	+	−	/
P7	Mean	9.34e+05	9.35e+05	9.33e+05	9.52e+05	9.33e+05	9.32e+05	9.32e+05	4.62e+06	4.17e+06	9.41e+05	9.34e+05	9.26e+05	2.57e+07	9.34e+05	9.32e+05
	STD	1.74e+03	1.58e+03	1.49e+03	3.83e+03	1.74e+03	1.55e+03	1.44e+03	6.15e+05	5.79e+05	1.53e+03	1.17e+03	6.24e+02	3.10e+02	4.62e+02	1.42e+03
	Best	9.31e+05	9.31e+05	9.30e+05	9.44e+05	9.30e+05	9.30e+05	9.30e+05	3.62e+06	3.30e+06	9.39e+05	9.32e+05	9.25e+05	2.57e+07	9.34e+05	9.30e+05
	Worst	9.37e+05	9.38e+05	9.35e+05	9.60e+05	9.38e+05	9.37e+05	9.36e+05	5.86e+06	5.77e+06	9.44e+05	9.37e+05	9.27e+05	2.57e+07	9.36e+05	9.36e+05
	p-value	3.18e−03	5.53e−08	8.77e−02	3.02e−11	3.26e−01	8.53e−01	8.30e−01	3.02e−11	3.02e−11	3.02e−11	7.22e−06	3.02e−11	3.02e−11	3.02e−11	/
	+/−/≈	+	+	≈	+	≈	≈	≈	+	+	+	+	−	+	−	/
P8	Mean	1.59e+01	1.43e+01	1.60e+01	1.83e+01	1.62e+01	1.60e+01	1.62e+01	2.14e+01	2.15e+01	1.37e+01	1.58e+01	1.55e+01	1.00e+30	1.51e+01	1.55e+01
	STD	5.93e−01	1.83e+00	6.12e−01	1.75e+00	7.28e−01	7.27e−01	5.48e−01	2.66e+00	1.69e+00	1.62e+00	5.08e−01	7.47e−01	1.43e+14	5.82e−01	7.00e−01
	Best	1.45e+01	1.04e+01	1.37e+01	1.56e+01	1.46e+01	1.44e+01	1.49e+01	1.42e+01	1.74e+01	8.69e+00	1.50e+01	1.28e+01	1.00e+30	1.42e+01	1.42e+01
	Worst	1.72e+01	1.83e+01	1.72e+01	2.18e+01	1.77e+01	1.73e+01	1.73e+01	2.56e+01	2.51e+01	1.64e+01	1.68e+01	1.67e+01	1.00e+30	1.61e+01	1.65e+01
	p-value	6.79e−02	3.67e−03	1.17e−02	4.62e−10	2.75e−03	5.32e−03	1.78e−04	6.52e−09	3.02e−11	8.84e−07	7.73e−02	4.14e−03	1.21e−12	3.40e−01	/
	+/−/≈	≈	−	+	+	+	+	+	+	+	−	≈	+	+	≈	/

Table 16
Summary of ADCCDE's performance compared with state-of-the-art DEs on real-world problems.

	jSO	LSHADE-cnEpSin	LSHADE-RSP	FODE	SCSS-L-SHADE	EaDE	DTDE	DE-EXP	ADE-DMRM	MSDE	ESADE	dFDB-LSHADE	mLSHADE-SPACMA	NSM-LSHADE-CnEpSin
W	5	5	7	6	4	3	6	6	5	5	5	4	7	3
L	0	2	0	0	0	0	1	1	1	3	0	3	1	2
T	3	1	1	2	4	5	1	1	2	0	3	1	0	3

Table 17
Overall performance ranking of the considered DEs on real-world problems.

Algorithm	Ranking
ADCCDE	**3.750**
dFDB-LSHADE	3.875
NSM-LSHADE-CnEpSin	5.250
ESADE	7.125
EaDE	7.375
DTDE	7.625
SCSS-L-SHADE	7.750
jSO	8.125
LSHADE-RSP	8.250
MSDE	8.500
LSHADE-cnEpSin	9.250
DE-EXP	9.875
ADE-DMRM	10.125
FODE	10.375
mLSHADE-SPACMA	12.750

achieving dynamic collaboration between the dual modules. To enable on-demand selection among the three, ADCC integrates population diversity and iteration stage information to realize fine-grained control throughout the entire process.

Different mutation strategies and algorithms are employed to validate the ADCC framework. Experimental results show that ADCC significantly improves the baseline DEs. The proposed ADCCDE algorithm achieves a statistically superior ranking compared to two baseline DE algorithms and several state-of-the-art DE variants. Furthermore, a comparison is conducted between the ADCC and ESA frameworks, verifying the effectiveness of dual-module dimension fusion.

The theoretical significance of the ADCCDE method lies in addressing the mismatch defect occurring in the instantaneous replacement of subpopulations. This defect emerges under the current paradigm, which features adaptive single subpopulation generation and continuous population states throughout the entire stage. By integrating the advantages of two complementary modules, collaborative solutions fused with the merits of multiple strategies are constructed and allowed to participate in competition. This provides a new way of thinking for solving such transition phase problems. Meanwhile, aiming at the problem that strategy mechanisms of most multi-strategy algorithms need to be preset in advance, a dynamically autonomous allocation mode is developed. This mode relies on the Euclidean distance between strategy offspring and parent individuals for allocation. It can reflect the dispersion degree of strategy subpopulations to a certain extent, realizing real-time allocation of current exploration and exploitation population modules. Thus, it offers a new analytical scheme for subsequent similar problems.

In future work, ADCCDE can be extended to more types of module combinations. These combinations are not limited to strategies and algorithms; the same idea may also be applied and extended to modules such as parameters. The high-level algorithm integration introduces relatively high complexity. In follow-up research, efforts can focus on the pre-utilizability of dynamic competition values. This enables advance prediction of the modules required for offspring generation, further improving the lightweight level of the algorithm. Meanwhile the algorithm can be integrated with the reward mechanism of reinforcement learning. Selecting optimal parameters in the current phase based on reinforcement learning to achieve parameter self-adaptation can effectively eliminate the impact of parameter sensitivity. Additionally, its applications in other types of optimization, such as noisy (Zhang et al., 2025), multi-modal (Yue et al., 2017), multi-objective (Lin et al., 2024) and multi-task (G. Li et al., 2022) optimizations, will be further expanded.

The source code of ADCCDE is available at https://zsxhomepage.github.io/.

CRediT authorship contribution statement

Ying Xia Wu: Writing – original draft, Validation, Software, Methodology. **Yu Hong Wang:** Writing – original draft, Validation, Software, Methodology. **Sheng Xin Zhang:** Writing – review & editing, Supervision, Funding acquisition, Conceptualization. **Li Ming Zheng:** Funding acquisition. **Shao Yong Zheng:** Supervision, Funding acquisition.

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.

Acknowledgments

The work described in this paper was supported by the National Natural Science Foundation of China under Grant 62201227 and Grant 62071503, the Research Grants Council (RGC) Hong Kong under Grant AoE/E-101/23-N, the National Special Project Number for International Cooperation under Grant 2015DFR11050, the Applied Science and Technology Research and Development Special Fund Project of Guangdong Province under Grant 2016B010126004 and the Guangzhou Basic and Applied Basic Research under Grant SL2022A04J01366.

Data availability

Data will be made available on request.

References

Abd El-Mageed, A.A., Abohany, A.A., Saad, H.M., Sallam, K.M., 2023. Parameter extraction of solar photovoltaic models using queuing search optimization and differential evolution. Appl. Soft Comput. 134, 110032.

Awad, N., Ali, M., Liang, J., Qu, B., Suganthan, P., 2016. Problem definitions and evaluation criteria for the CEC 2017 special session and competition on single objective bound constrained real-parameter numerical optimization. In: Technical Report. Nanyang Technological University Singapore Singapore, pp. 1–34.

Awad, N.H., Ali, M.Z., Qu, B.Y., Liang, J.J., Suganthan, P.N., Nanyang Technological University, Singapore, Tech. Rep. 2016. Problem Definitions and evaluation criteria for the CEC 2017 special session and competition on single objective bound constrained Real-Parameter Numerical Optimization.

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, pp. 372–379.

Biswas, P.P., Suganthan, P.N., Wu, G., Amaratunga, G.A., 2019a. Parameter estimation of solar cells using datasheet information with the application of an adaptive differential evolution algorithm. Renew. Energy 132, 425–438.

Biswas, P.P., Suganthan, P.N., Wu, G., Amaratunga, G.A., 2019b. Parameter estimation of solar cells using datasheet information with the application of an adaptive differential evolution algorithm. Renew. Energy 132, 425–438.

Brest, J., Maučec, M., Bošković, B., 2017. Single objective real-parameter optimization: algorithm jSO. In: 2017 IEEE Congress on Evolutionary Computation. CEC, pp. 1311–1318.

Bujok, P., Kolenovsky, P., 2022. Eigen crossover in cooperative model of evolutionary algorithms applied to CEC 2022 single objective numerical optimisation. In: 2022 IEEE Congress on Evolutionary Computation. CEC, IEEE, pp. 1–8.
Cai, Z., Gao, S., Yang, X., Zhou, M., 2024. Multiselection-based differential evolution. IEEE Trans. Syst. Man Cybern.: Syst..
Cao, Z., Xu, K., Jia, H., Fu, Y., Foh, C.H., Tian, F., 2025. An autonomous differential evolution based on reinforcement learning for cooperative countermeasures of unmanned aerial vehicles. Appl. Soft Comput. (ISSN: 1568-4946) 169, 112605.
Chen, B., Ouyang, H., Li, S., Ding, W., 2025. Dual-stage self-adaptive differential evolution with complementary and ensemble mutation strategies. Swarm Evol. Comput. 93, 101855.
Chen, B., Ouyang, H., Li, S., Zou, D., 2024. Differential evolution algorithm with a complementary mutation strategy and data fusion-based parameter adaptation. Inform. Sci. 668, 120522.
Cleophas, T.J., Zwinderman, A.H., 2016. Non-parametric tests for three or more samples (Friedman and Kruskal-Wallis). In: Clinical Data Analysis on a Pocket Calculator: Understanding the Scientific Methods of Statistical Reasoning and Hypothesis Testing. Springer, pp. 193–197.
Das, S., Suganthan, P.N., 2010. Problem definitions and evaluation criteria for CEC 2011 competition on testing evolutionary algorithms on real world optimization problems. Jadavpur Univ. Nanyang Technol. Univ. Kolkata 341–359.
Fu, S., Ma, C., Li, K., Xie, C., Fan, Q., Huang, H., Xie, J., Zhang, G., Yu, M., 2025. Modified LSHADE-SPACMA with new mutation strategy and external archive mechanism for numerical optimization and point cloud registration. Artif. Intell. Rev. 58 (3), 72.
Gao, S., Yu, Y., Wang, Y., Wang, J., Cheng, J., Zhou, M., 2019. Chaotic local search-based differential evolution algorithms for optimization. IEEE Trans. Syst. Man Cybern.: Syst. 51 (6), 3954–3967.
Guo, H., Ma, Y., Ma, Z., Chen, J., Zhang, X., Cao, Z., Zhang, J., Gong, Y.-J., 2024. Deep reinforcement learning for dynamic algorithm selection: A proof-of-principle study on differential evolution. IEEE Trans. Syst. Man Cybern.: Syst. 54 (7), 4247–4259.
Gupta, S., Singh, S., Su, R., Gao, S., Bansal, J.C., 2023. Multiple elite individual guided piecewise search-based differential evolution. IEEE/CAA J. Autom. Sin. 10 (1), 135–158.
Huang, C., Zhu, J., Xu, M., 2025. DyS-MPADE: A novel multipopulation adaptive differential evolution methodology based on dynamic subpopulation. J. Comput. Des. Eng. 12 (3), 204–225.
Jangir, P., Arpita, Agrawal, S.P., Pandya, S.B., Parmar, A., Kumar, S., Tejani, G.G., Abualigah, L., 2025. A cooperative strategy-based differential evolution algorithm for robust PEM fuel cell parameter estimation. Ionics 31 (1), 703–741.
Jin, X.L., Zhang, S.X., Zheng, L.M., Zheng, S.Y., 2025. Differential evolution algorithm with local and global parameter adaptation. Swarm Evol. Comput. 98, 102125.
Kaur, M., Gianey, H.K., Singh, D., Sabharwal, M., 2019. Multi-objective differential evolution based random forest for e-health applications. Modern Phys. Lett. B 33 (05), 1950022.
Li, J.-Y., Du, K.-J., Zhan, Z.-H., Wang, H., Zhang, J., 2022. Distributed differential evolution with adaptive resource allocation. IEEE Trans. Cybern. 53 (5), 2791–2804.
Li, S., Li, W., Tang, J., Wang, F., 2023. A new evolving operator selector by using fitness landscape in differential evolution algorithm. Inform. Sci. 624, 709–731.
Li, W., Meng, X., Huang, Y., 2021. Fitness distance correlation and mixed search strategy for differential evolution. Neurocomputing 458, 514–525.
Li, X., Wang, L., Jiang, Q., Li, N., 2021. Differential evolution algorithm with multi-population cooperation and multi-strategy integration. Neurocomputing 421, 285–302.
Li, G., Zhang, Q., Wang, Z., 2022. Evolutionary competitive multitasking optimization. IEEE Trans. Evol. Comput. 26 (2), 278–289.
Liao, Z., Pang, Q., Gu, Q., 2024. Differential evolution based on strategy adaptation and deep reinforcement learning for multimodal optimization problems. Swarm Evol. Comput. 87, 101568.
Lin, X., Meng, Z., 2024. An adaptative differential evolution with enhanced diversity and restart mechanism. Expert Syst. Appl. 249, 123634.
Lin, J., Zhang, S.X., Xu, Y.J., Zheng, S.Y., 2024. A hypervolume fraction-based adaptive evolutionary algorithm for many-objective optimization and the application to electromagnetic device design. Eng. Appl. Artif. Intell. 137, 109060.
Meng, Z., Chen, Y., 2023. Differential evolution with exponential crossover can be also competitive on numerical optimization. Appl. Soft Comput. 146, 110750.
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.
Nadimi-Shahraki, M.H., Zamani, H., 2022. DMDE: Diversity-maintained multi-trial vector differential evolution algorithm for non-decomposition large-scale global optimization. Expert Syst. Appl. 198, 116895.
Öztürk, H.T., Kahraman, H.T., 2025. Metaheuristic search algorithms in frequency constrained truss problems: Four improved evolutionary algorithms, optimal solutions and stability analysis. Appl. Soft Comput. 171, 112854.
Price, K.V., Storn, R.M., Lampinen, J.A., 2005. Differential Evolution: A Practical Approach to Global Optimization. Springer.
Qiao, K., Liang, J., Yu, K., Yuan, M., Qu, B., Yue, C., 2022. Self-adaptive resources allocation-based differential evolution for constrained evolutionary optimization. Knowl.-Based Syst. 235, 107653.
Qin, A.K., Huang, V.L., Suganthan, P.N., 2008. Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Trans. Evol. Comput. 13 (2), 398–417.
Reyes-Davila, E., Haro, E.H., Casas-Ordaz, A., Oliva, D., Avalos, O., 2025. Differential evolution: A survey on their operators and variants. Arch. Comput. Methods Eng. 32 (1), 83–112.
Sheskin, D.J., 2003. Handbook of Parametric and Nonparametric Statistical Procedures. Chapman and Hall/CRC.
Srivastava, G., Pradhan, N., 2023. Handling imbalanced class in melanoma: Kemeny–Young rule based optimal rank aggregation and Self-Adaptive Differential Evolution Optimization. Eng. Appl. Artif. Intell. 125, 106738.
Stanovov, V., Akhmedova, S., Semenkin, E., 2018. LSHADE algorithm with rank-based selective pressure strategy for solving CEC 2017 benchmark problems. In: 2018 IEEE Congress on Evolutionary Computation. CEC, pp. 1–8.
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, G., Yang, B., Yang, Z., Xu, G., 2020. An adaptive differential evolution with combined strategy for global numerical optimization. Soft Comput. 24 (9), 6277–6296.
Tan, Z., Tang, Y., Li, K., Huang, H., Luo, S., 2022. Differential evolution with hybrid parameters and mutation strategies based on reinforcement learning. Swarm Evol. Comput. 75, 101194.
Tang, L., Dong, Y., Liu, J., 2014. Differential evolution with an individual-dependent mechanism. IEEE Trans. Evol. Comput. 19 (4), 560–574.
Tang, J., Geng, L., Pang, J., Fu, J., Pu, M., 2026. FLADE: Guiding differential evolution through fitness landscape sequence analysis for influence maximization. Expert Syst. Appl. (ISSN: 0957-4174) 299, 130229.
Tang, G., Yao, W., Li, C., Jiang, T., Yang, S., 2024. Black-box adversarial patch attacks using differential evolution against aerial imagery object detectors. Eng. Appl. Artif. Intell. 137, 109141.
Tatsis, V.A., Parsopoulos, K.E., 2023. Reinforcement learning for enhanced online gradient-based parameter adaptation in metaheuristics. Swarm Evol. Comput. 83, 101371.
Wang, K., Gao, S., Zhou, M., Zhan, Z.-H., Cheng, J., 2024. Fractional order differential evolution. IEEE Trans. Evol. Comput..
Wang, M., Ma, Y., 2023. A differential evolution algorithm based on accompanying population and piecewise evolution strategy. Appl. Soft Comput. 143, 110390.
Wang, Y., Yu, X., Zhang, W., 2025. An improved reinforcement learning-based differential evolution algorithm for combined economic and emission dispatch problems. Eng. Appl. Artif. Intell. (ISSN: 0952-1976) 140, 109709.
Wolpert, D.H., Macready, W.G., 1997. No free lunch theorems for optimization. IEEE Trans. Evol. Comput. 1 (1), 67–82.
Yi, W., Chen, Y., Pei, Z., Lu, J., 2022. Adaptive differential evolution with ensembling operators for continuous optimization problems. Swarm Evol. Comput. 69, 100994.
Yildirim, İ.b., Bozkurt, M.H., Kahraman, H.T., Aras, S., 2025. Dental X-Ray image enhancement using a novel evolutionary optimization algorithm. Eng. Appl. Artif. Intell. 142, 109879.
Yu, X., Wu, Z., Zhang, H., Li, Y., 2025. Wind farm layout optimization through multi-strategy differential evolution-based reinforcement learning algorithm. Energy (ISSN: 0360-5442) 335, 138221.
Yue, C., Qu, B., Liang, J., 2017. A multiobjective particle swarm optimizer using ring topology for solving multimodal multiobjective problems. IEEE Trans. Evol. Comput. 22 (5), 805–817.
Zhang, S.X., Chan, W.S., Peng, Z.K., Zheng, S.Y., Tang, K.S., 2020. Selective-candidate framework with similarity selection rule for evolutionary optimization. Swarm Evol. Comput. 56, 100696.
Zhang, S.X., Chan, W.S., Tang, K.S., Zheng, S.Y., 2021. Adaptive strategy in differential evolution via explicit exploitation and exploration controls. Appl. Soft Comput. 107, 107494.
Zhang, S.X., Hu, X.R., Zheng, S.Y., 2024. Differential evolution with evolutionary scale adaptation. Swarm Evol. Comput. 85, 101481.
Zhang, S.X., Liu, Y.H., Hu, X.R., Zheng, L.M., Zheng, S.Y., 2025. Fully Informed Fuzzy Logic System Assisted Adaptive Differential Evolution Algorithm for Noisy Optimization. IEEE Trans. Fuzzy Syst. 33 (6), 1876–1888.
Zhang, J., Sanderson, A.C., 2009. JADE: adaptive differential evolution with optional external archive. IEEE Trans. Evol. Comput. 13 (5), 945–958.
Zhang, S.X., Wen, Y.N., Liu, Y.H., Zheng, L.M., Zheng, S.Y., 2023. Differential evolution with domain transform. IEEE Trans. Evol. Comput. 27, 1440–1455.
Zhang, S.X., Zheng, S.Y., Zheng, L.M., 2017. An efficient multiple variants coordination framework for differential evolution. IEEE Trans. Cybern. 47 (9), 2780–2793.
Zheng, L.M., Zhang, S.X., Tang, K.S., Zheng, S.Y., 2017. Differential evolution powered by collective information. Inform. Sci. 399, 13–29.