Complex & Intelligent Systems (2024) 10:525–550
https://doi.org/10.1007/s40747-023-01162-9

ORIGINAL ARTICLE

An integrated differential evolution of multi-population based on contribution degree

Yufeng Wang[1] · Hao Yang[1] · Chunyu Xu[2,3] · Yunjie Zeng[1] · Guoqing Xu[4]

Received: 19 December 2022 / Accepted: 25 June 2023 / Published online: 26 July 2023

Abstract
The differential evolution algorithm based on multi-population mainly improves its performance through mutation strategy and grouping mechanism. However, each sub-population plays a different role in different periods of iterative evolution. If each sub-population is assigned the same computing resources, it will waste a lot of computing resources. In order to rationally distribute computational resources, an integrated differential evolution of multi-population based on contribution degree (MDE-ctd) is put forth in this work. In MDE-ctd, the whole population is divided into three sub-populations according to different update strategies: archival, exploratory, and integrated sub-populations. MDE-ctd dynamically adjusts computing resources according to the contribution degree of each sub-population. It can effectively use computing resources and speed up convergence. In the updating process of integrated sub-populations, a mutation strategy pool and two-parameter value pools are used to maintain population diversity. The experimental results of CEC2005 and CEC2014 benchmark functions show that MDE-ctd outperforms other state-of-art differential evolution algorithms based on multi-population, especially when it deals with highly complex optimization problems.

✉ Hao Yang
yhao0908@163.com

[1] School of Software and Computer Science, Nanyang Institute of Technology, Henan 473000, China

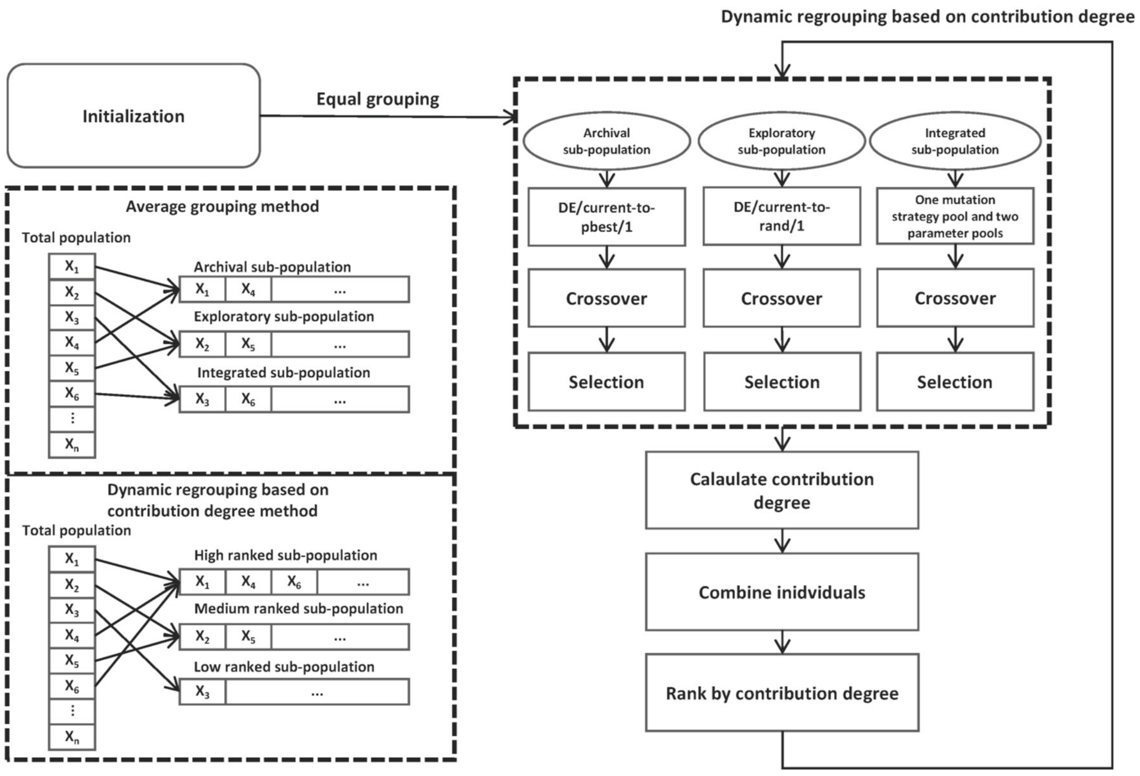
[2] Electronic Information School, Wuhan University, Hubei 430072, China

[3] School of Information Engineering, Nanyang Institute of Technology, Henan 473000, China

[4] School of Electronic Information Engineering, Henan Polytechnic Institute, Henan 473000, China

Graphical abstract

An integrated differential evolution of multi-population based on contribution degree


Dynamic regrouping based on contribution degree
Initialization
Equal grouping
Archival sub-population
Exploratory sub-population
Integrated sub-population
DE/current-to-pbest/1
DE/current-to-rand/1
One mutation strategy pool and two parameter pools
Crossover
Selection
Average grouping method
Total population
Archival sub-population
Exploratory sub-population
Integrated sub-population
Calaulate contribution degree
Combine inidviduals
Rank by contribution degree
Dynamic regrouping based on contribution degree method
High ranked sub-population
Medium ranked sub-population
Low ranked sub-population

Keywords Differential evolution · Contribution degree · Multi-population · Dynamic regrouping

Introduction

The Differential Evolution (DE) algorithm was first proposed by Storn and Price in 1995 [1], it is widely used to solve real number optimization problems. DE is an adaptive global optimization algorithm based on population. It solves optimization problems through the iteration of evolutionary operators, and its steps include mutation, crossover and selection [2, 3]. Because of its simple structure and easy implementation, it is widely used in data mining, artificial neural networks, digital filter design, electromagnetism and other fields. For example, solar cell and module parameter estimation [4], Electromagnetic framework [5], Environmental and economic power dispatching [6], protein structure prediction [7].

Different from other algorithms [8, 9], DE mainly balances the trade-off of exploration and exploitation through mutation strategy, population size NP, scaling factor F, crossover rate CR and other parameters. In the early stage, DE changes its parameters and mutation strategy through trial and error in order to obtain the best result. However, this approach necessitates a lot of testing, therefore many researchers started looking into adaptive ways to increase effectiveness and conserve computing resources. Adaptive technology is introduced to dynamically update the control parameters to optimize the problem. Additionally, if the parameter adaptive design is suitable, the algorithm's convergence performance can be enhanced.

According to the classification scheme introduced by Angeline [10] and Eiben et al. [11, 12], parameter control mechanisms can be divided into the following three categories:

1. Deterministic rule setting parameters. In an evolutionary iteration, the setting parameters are controlled by some deterministic rules and does not accept any feedback information. For instance, the mutation rate parameter is changed by the time-dependent [13].
2. Adaptive setting parameters. In the search process, setting parameters are dynamically changed by interacting feedback information. The adaptive setting parameter set has a great advantage in improving the convergence rate and global optimal search probability of the algorithm. This category includes many recently suggested DE algorithms, including jDE [14], SaDE [15], JADE [16], SHADE [17], and others.

3. Self-adaptive setting parameters. The setting parameters can be adaptively changed in the process of evolution. Better parameter values can be passed on to more offspring because they are more likely to generate better individuals. In this category, SPDE [18] and DESAP [19] both fit.

In addition to adaptive technology, the prevalence of distributed computing offers a fresh method for enhancing the performance and accessibility of algorithms. The population is put on a distributed system by the researchers to enhance the performance of the DE method. Due to the population's scattered structure, it is possible for each sub-population member to find an optimal solution independently during evolution. An adaptive DDE (ADDE) is proposed by Zhan et al. [20]. In ADDE, the total population is divided into four sub-populations, including one master population and three slave populations, using a master-slave multi-population distribution framework. The master population is responsible for collecting individuals from the three slave populations and reassigning them to the three slave populations after distinguishing them, while different slave populations will adaptively choose their own mutation strategies according to the feedback of individuals. Wu et al. propose a multi-population based framework (MPF) in EDEV [21] to achieve the purpose of integrating multiple DE variants. In EDEV, each DE variant evolves independently, and computing resources are dynamically allocated to the DE variants based on their performance. Li et al. introduce an information-sharing mechanism in the sub-population of MPMSDE [22] to avoid falling into local optima. In Cloudde [23], the author divides the population into four parallel sub-populations and uses different mutation strategies in the concurrent sub-populations to reduce the calculation cost of practical problems and meet different search requirements.

A multi-population based technique is used in MPEDE [24] presented by Wu et al. It can achieve the dynamic integration of numerous mutation strategies. A reward sub-population and three other sub-populations are created at MPEDE, with the reward sub-population being assigned to one of the other sub-population. The three sub-populations adopt different mutation methods and different updating techniques, respectively, which include "DE/current-to-pbest/1" and archive, "DE/current-to-rand/1" and "DE/rand /1" strategies. A strategy with the highest recent performance will automatically receive additional computing resources as the algorithm develops, while the remaining computing resources will be distributed equally among the other two strategies.

However, there are still some problems with these works. The greatest computational resources are made available to the best mutant strategy in MPEDE throughout evolution, whereas the computational resources of medium and lower strategies are equal. This allocation strategy saves computing resources. The medium strategy should have more computational resources than the inferior strategy in order to utilize them more effectively. The "DE/rand/1" mutation technique provides an advantage in increasing population diversity, but it may reduce the rate of convergence [25]. The "DE/current-to-pbest/1" method in MPEDE employs the arithmetic mean to control parameters, which could result in premature convergence of the algorithm [26].

In order to address these issues, an integrated differential evolution of multi-population based on contribution degree (MDE-ctd) is proposed. It employs a new grouping technique (Dynamic regrouping method, DRM) to balance the distribution of computing resources. DRM can calculate the contribution degree of three sub-populations, and regroup the size of the three sub-populations according to the rank of contribution degree. In the process of evolution, integrated sub-population expanded the search space and lessens the impact of strategy on convergence speed by using one mutation strategy pool("DE/best/2", "DE/rand/1" and "DE/current-to-rand/1") and two-parameter value pools(CR parameter pool and F parameter pool). Additionally, historical successful weight (HSW) parameter adaptive approaches were used for "DE/current-to-pbest/1" parameters. The adaptive technology can avoid premature convergence, speeds up convergence and avoids local optimization. On the benchmark function set of CEC 2005 and CEC 2014, MDE-ctd was tested in several dimensions, and an exhaustive comparison with a number of state-of-the-art DE variants fully illustrated its competitive performance. The MDE-ctd algorithm performed well, especially for handling highly complex optimization issues.

Related work

This section discusses computing resource allocation methods, the traditional DE, differential evolution of multi-population based on the ensemble of mutation strategies (MPEDE) [24], and Ensemble of differential evolution variants (EDEV) [21].

Computational resource allocation methods

In the process of evolution, a single mutation strategy is very different in the face of multi-modal, constrained, large-scale, dynamic, and uncertain optimization problems, which cannot make the algorithm achieve global optimality. Many researchers began to use multiple mutation strategies to co-evolve. In addition, effective computing resource allocation methods are also valued by many researchers. Reasonable computing resource allocation methods can not only avoid the waste of computing resources but also maintain the good search performance of the algorithm. The use of these two methods is not limited to some DE algorithms mentioned in the first section. In other studies such as particle swarm optimization and coevolution, there are also contribution-based computing resource allocation methods and multiple population strategies.

In MPCPSO [27], Li et al. divided the whole population into "elite group (EP)" and "general group (GP)", and each group adopted different learning strategies. By using a dynamic segment-based average learning strategy (DSMLS) and multi-dimensional comprehensive learning strategy (MDCLS) simultaneously, information sharing and simultaneous evolution among populations can be realized. In DCCA [28], a two-layer distributed coevolutionary structure is proposed. On the one hand, the scalability of dimension division is obtained, and high-dimensional problems are effectively handled. On the other hand, the subcomponents can quickly adapt to the changes in resource allocation, and achieve an effective allocation of computing resources. In DCCC [29], a cooperative coevolutionary framework based on difficulty and contribution is proposed, which allocates more computing resources to sub-problems with greater and more difficult contributions, thus comprehensively solving the problem of difficulty and imbalance of computing resources. In the CBCCO [30] proposed by Jia et al., the contribution-based overlapping problem decomposition (CBD) method is used to allocate computing resources to the sub-components with larger contributions, effectively and efficiently decompose and optimize the non-separable large problems with overlapping sub-components.

DE algorithm

DE is a population-based optimization method, which is a kind of evolutionary algorithm. The main difference between DE and other evolutionary algorithms (EA) is that the DE algorithm is based on individual differences [31]. Evolution is composed of four basic steps: initialization, mutation, crossover and selection [32]. The DE implementation process is as follows.

Initialization

The population is randomly created during the initialization phase in accordance with the uniform distribution in the search space as:

$$x_{i,j,0} = L_j + \text{rand} \times (U_j - L_j) \tag{1}$$

where i is the individual index, U_j and L_j are upper bounds of the jth dimension and lower bounds of the jth dimension. rand is a random number in the range $[0, 1]$. Once the operator has been initialized, DE optimizes it via a sequence of evolutions that include mutation, crossover, and selection until the termination condition is met.

Mutation

At g generation, for each individual $x_{i,g}$ in the current population, there have some mutants based on the mutation strategy as $\{v_{i,g} = (v_{i,1,g}, v_{i,2,g}, \ldots, v_{i,D,g}) \mid i = 1, 2, 3, \ldots, \text{NP}\}$, where NP is the population size, D is the dimension of the problem. The following are some common mutation strategies in the literature:

DE/rand/1:

$$v_{i,g} = x_{r1,g} + F \cdot (x_{r2,g} - x_{r3,g}) \tag{2}$$

DE/rand/2:

$$v_{i,g} = x_{r5,g} + F \cdot (x_{r1,g} - x_{r2,g}) + F \cdot (x_{r3,g} - x_{r4,g}) \tag{3}$$

DE/current-to-best/1:

$$v_{i,g} = x_{i,g} + F \cdot (x_{\text{best},g} - x_{i,g}) + F \cdot (x_{r1,g} - x_{r2,g}) \tag{4}$$

DE/current-to-rand/1:

$$v_{i,g} = x_{i,g} + F \cdot (x_{r3,g} - x_{i,g}) + F \cdot (x_{r1,g} - x_{r2,g}) \tag{5}$$

DE/current-to-pbest/1:

$$v_{i,g} = x_{i,g} + F \cdot (x^{p}_{\text{best},g} - x_{i,g}) + F \cdot (x_{r1,g} - x_{r2,g}) \tag{6}$$

where r_1, r_2, r_3, r_4 and r_5 are mutually distinct random values drawn from the range $[1, \text{NP}]$. The best individual in the g-generation of the population is $X_{\text{best},g}$. From the present population's best p percent, $X^{p}_{\text{best},g}$ is chosen at random. F is used to scale the difference vector in order to control the search step.

Crossover

After the mutation operator, the initial vector is crossed with the mutation vector to generate the target vector of the test vectors $u_{i,g} = (u_{i,1,g}, u_{i,2,g}, \ldots, u_{i,D,g})$. In DE, crossover operation can be implemented by using one of three methods: binary crossover, exponential crossover [33], and arithmetic crossover. DE usually uses a binomial crossover defined as :

$$u_{i,j,g} = \begin{cases} v_{i,j,g}, & \text{if } (\text{rand} \le \text{CR } \mathit{or} \ j = j_{\text{rand}}) \\ x_{i,j,g}, & \text{otherwise} \end{cases} \tag{7}$$

where j_{rand}= RandInt(1, D) is a random integer from 1 to D, it can make sure that at least one variable of the trial vector $u_{i,g}$ comes from the mutation vector $v_{i,g}$. CR is the crossover rate, if CR =1, there is no crossover, and the test vector is equal to the mutation vector. CR determines the proportion of $u_{i,g}$ inherited from $v_{i,g}$.

Selection

If the value of the newly generated decision variable is greater than the upper bound or less than the lower bound, it will be reset to the corresponding bound and continuously re-initialized within the predetermined range. Then, the selection operation is performed after evaluating the target function values of all test vectors.

In order to maintain a better vector in the population, the fitness of the trial vector is calculated after the crossover operator, and the selection operator is executed. The fitness values of $x_{i,g}$ and $u_{i,g}$ are used to determine which of $x_{i,g}$ or $u_{i,g}$ will survive in the next generation. For the minimization problem, vectors with lower fitness values always advance to the following generation, as illustrated by

$$x_{i,g+1} = \begin{cases} u_{i,g}, & \text{if } f(u_{i,g}) \le f(x_{i,g}) \\ x_{i,g}, & \text{otherwise} \end{cases} \tag{8}$$

where f(x) is the fitness evaluation function.

MPEDE

MPEDE [24] is a multi-population-based differential evolution algorithm, which uses three different mutation strategies. The entire population is dynamically divided into four populations, including three smaller sub-populations and a larger reward population. Three mutation strategies are chosen by MPEDE: "DE/current-to-rand/1" and archive, "DE/current-to-rand/1" and "DE/rand/1". In the process of evolution, the reward population is dynamically allocated to the dominant population according to the proportion of population fitness improvement of different strategies.

It is crucial to choose the right control parameters for each mutation strategy because the performance of the DE algorithm and the mutation strategy is correlated. As a result, different mutation strategies may require different parameter settings.

MPEDE uses adaptive parameter technology. the scale factor $F_{i,j}$ can help individual x_i using jth mutation strategy to generate the crossover probability of the trial solution. The scale factor $F_{i,j}$ is generated from the Cauchy distribution of the position parameter μF_j and the scale parameter 0.1. The crossover probability $\mathrm{CR}_{i,j}$ is the crossover probability of the trial solution generated by the jth mutation strategy for individual x_i. The crossover probability is generated by the normal distribution of mean $\mu\mathrm{CR}_j$ and standard variance 0.1. In each generation of g, the parameter set formula is as follows:

$$\begin{cases} F_{ij} = \mathrm{rand}c_{i,j}(\mu F_j, 0.1) \\ \mathrm{CR}_{i,j} = \mathrm{rand}n_{i,j}(\mu \mathrm{CR}_j, 0.1) \end{cases} \tag{9}$$

$$\mu F_j = (1-c)\cdot \mu F_j + c \cdot \mathrm{mean}_A(S_{F,j}) \tag{10}$$

$$\mu \mathrm{CR}_j = (1-c)\cdot \mu \mathrm{CR}_j + c \cdot \mathrm{mean}_L(S_{\mathrm{CR},j}) \tag{11}$$

where c is a positive constant between 0 and 1, $\mathrm{mean}_A(S_{F,j})$ is an arithmetic mean value of $S_{F,j}$ [26], and $\mathrm{mean}_L(S_{\mathrm{CR},j})$ is the Lehmer mean of $\mathrm{CR}_{F,j}$ [17] which are defined as

$$\mathrm{mean}_A(S_{F,j}) = \frac{1}{m}\sum_{k=1}^{m} S_{F,k} \tag{12}$$

$$\mathrm{mean}_L(S_{\mathrm{CR},j}) = \frac{\sum_{k=1}^{|S_{\mathrm{CR}}|} S_{\mathrm{CR},k}^2}{\sum_{k=1}^{|S_{\mathrm{CR}}|} S_{\mathrm{CR},k}} \tag{13}$$

EDEV

Wu et al. propose in 2017 that EDEV is a practical multi-population based framework (MPF) [21] for integrating multiple DE variants. EDEV is composed of three traditional DE versions: JADE [16], CoDE [34], and EPSDE [35]. In the evolutionary process, each component DE variable in EDEV is assigned to an indicator sub-population using the MPEDE allocation method. After a certain number of iterations, the reward sub-population is assigned to the DE variant with the best performance recently. The following three DE variants that are utilized in EDEV are briefly described here.

JADE

The "DE/current-to-pbest/1" mutation approach is employed in JADE, and it consists of two versions: one has archive and the other does not. The archival version uses the historical optimal parameters generated by a normal distribution and Cauchy distribution to realize the self-adaptation of parameters F and CR.

CoDE

Three alternative mutation strategies are utilized in CoDE: "DE/rand/1", "DE/rand/2", and "DE/current-to-rand/1". Additionally, there are three control parameter combinations of $[F = 1.0, \mathrm{CR} = 0.1]$, $[F = 1.0, \mathrm{CR} = 0.9]$, and $[F = 0.8, \mathrm{CR} = 0.2]$. Each parent vector generation of the iterative process of evolution uses three alternative mutation techniques to produce child vectors, and if the child vector produces a result that is better than the parent vector, the child vector would continue to exist.

EPSDE

In EPSDE, the "DE/best/2", "DE/rand/1", and "DE/current-to-rand/1" mutation strategy pools are present. The CR parameter pool value range is [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9], while the F parameter pool value range is [0.4, 0.5, 0.6, 0.7, 0.8, 0.9]. A mutation method and associated parameter value are randomly allocated to each member of the initial population of their respective pools. The related mutation strategies and parameter values are permitted to survive when the child vector produced during the process of evolution outperforms the parent vector, while when the child vector generated in the evolution process cannot be superior to the parent vector, the mutation strategy and parameter values are randomly reinitialized.

An integrated differential evolution of multi-population

The traditional DE based on multi-population allocates computing resources of the same size to each sub-population, which is not conducive to the full and rational use of computing resources. MDE-ctd divides the whole population into three sub-populations (archival sub-population, exploratory sub-population and integrated sub-population.) based on mutation strategy. In the iterative evolution process, the contribution degree of each sub-population is used to rank, and the computing resources are dynamically adjusted according to the ranking results. The main framework of MDE-ctd was shown in Fig. 1.

In MDE-ctd, the mutation strategy of the archival sub-population is "DE/current-to-pbest/1", the mutation strategy of the exploratory sub-population is "DE/current-to-rand/1", and the integrated sub-population uses one mutation strategy pool ("DE/best/2", "DE/rand/1" and "DE/current-to-rand/1") and two-parameter value pools (CR parameter pool [0.1–0.9] and F parameter pool [0.4–0.9]).

Contribution degree

The contribution degree is the participation degree of each sub-population in finding the global optimal solution, it can count the proportion of the optimal fitness of each sub-population in the whole population.

The sub-population based on mutation strategy is ranked according to the ratio between the fitness change value of the previous ng generation of the sub-population and the evaluation of the consumption function. The indicators of the high-ranked sub-population h and the low-ranked sub-population l are defined as follows:

$$h = \arg\left(\max_{1\le j\le 3}\left(\frac{\Delta f_j}{ng\cdot \mathrm{NP}_j}\right)\right) \tag{14}$$

$$l = \arg\left(\min_{1\le j\le 3}\left(\frac{\Delta f_j}{ng\cdot \mathrm{NP}_j}\right)\right) \tag{15}$$

Where NP_j is the size of the jth sub-population, Δf_j is the accumulation of fitness change value brought by the jth mutation strategy in the previous generation, and $ng\cdot \mathrm{NP}_j$ is the function evaluations consumed of the jth mutation strategy in the previous ng generation. The order of the medium mutation strategy except for the best mutation strategy h and the worst mutation strategy l is m.

After ranking the sub-population, The high-ranked sub-population h received 3 points, the medium-ranked sub-population m with 2 points, and the low-ranked sub-population l with 1 point. The size of the sub-population can be determined as follows.

$$\lambda_j = \frac{\mathrm{score}_j}{\sum_{j=1,\ldots 3}\mathrm{score}_j} \tag{16}$$

$$\mathrm{NP}_j^{new} = \lambda_j \cdot \mathrm{NP}, \quad j = 1,\ldots,3 \tag{17}$$

Where score_j is the score of jth strategy, $\lambda_j$is the proportion of the jth strategy contribution, and NPis the total population size.

For different types of optimization problems, MDE-ctd can dynamically allocate computing resources according to the contribution degree. It can make more computing resources available to mutation strategies that are suitable for this type of optimization problem, and allocate less computing resources to the poor strategy, thus making full use of computing resources.

Multi-population and multi-strategy integration method

In the whole iterative process of DE, the mutation strategy plays different roles in different periods. In the early stage, individuals are scattered. In the middle and late stages, individuals will gather in the local or global optimal regions.

In MDE-ctd, the archival sub-population uses the "DE/current-to-pbest/1 (with achieve)" mutation strategy, it can preserve good search performance, accelerate convergence, and improve local search ability. The "DE/current-to-rand/1" method was used in the exploratory sub-population to allow individuals to explore more territory and prevent populations from clumping together too early in the evolutionary process. Note that the "DE/current-to-rand/1" strategy does not use crossover.

The mutation strategy and the set of parameter values are used in the renewal process of the integrated sub-population.

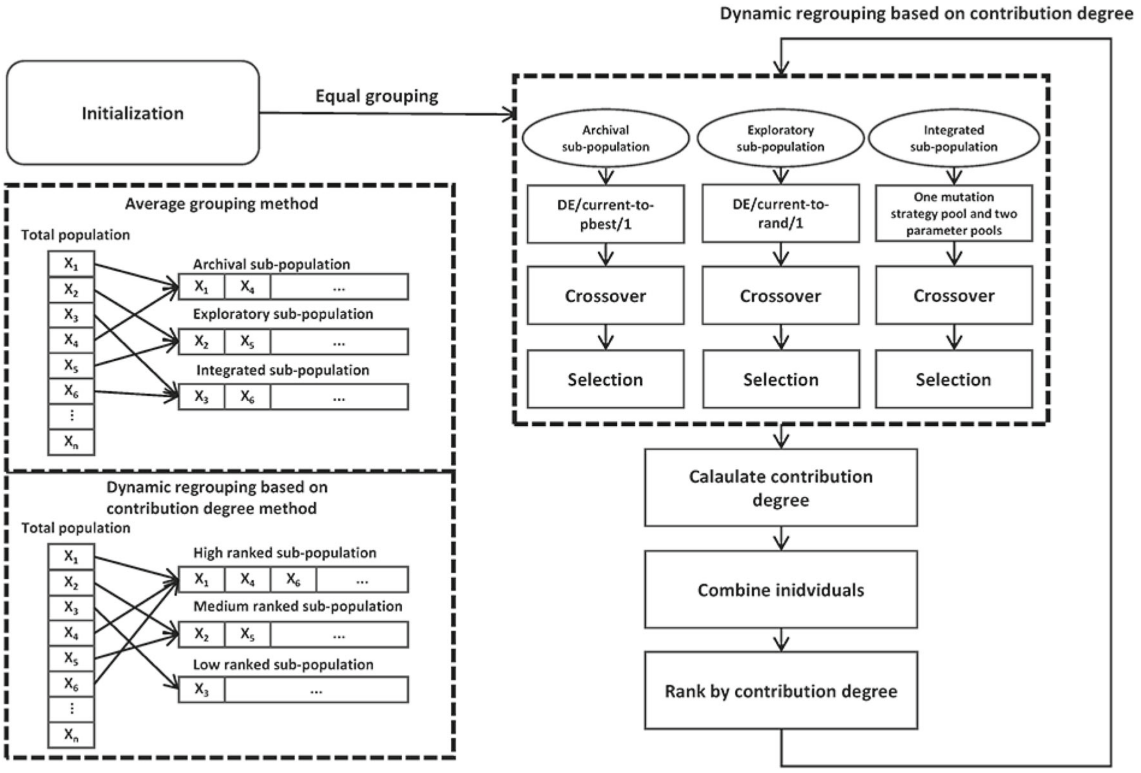

Dynamic regrouping based on contribution degree
Initialization
Equal grouping
Archival sub-population
Exploratory sub-population
Integrated sub-population
DE/current-to-pbest/1
DE/current-to-rand/1
One mutation strategy pool and two parameter pools
Crossover
Selection
Average grouping method
Total population
Archival sub-population
Exploratory sub-population
Integrated sub-population
Dynamic regrouping based on contribution degree method
High ranked sub-population
Medium ranked sub-population
Low ranked sub-population
Calaulate contribution degree
Combine inidviduals
Rank by contribution degree

Fig. 1 The main framework of MDE-ctd

It had a pool of values for each relevant parameter as well as a pool of mutation strategies. When tackling various real number optimization challenges, these various mutation methods were employed to compete to develop offspring populations that can exhibit various performance traits at various phases of evolution. The collection contained the mutation techniques "DE/rand/1," "DE/best/2", and "DE/current-to-rand/1". In a numerical pool, the parameters F and CR, $F \in [0.4, 0.5, 0.6, 0.7, 0.8, 0.9]$, CR $\in [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]$. Each member of the integrated sub-population was assigned a mutation strategy and randomly selected parameter values from their respective pools.

Adaptive parameter settings based on historical successful weight

Effective parameter combinations (F and CR) can improve the performance of DE. It is desirable to record and reuse these individual characteristics in later generations, thus enabling excellent individuals to have a higher chance of survival. However, different mutation strategy requires different combinations of parameters.

In order to use different parameter combinations rationally, an improved parameter adaptive method based on historical successful weight (HSW) was designed to dynamically update F and CR. JADE uses all successful F and CR values to guide individual parameter adjustment, while the core idea of HSW is to judge the advantages of contemporary weights by whether it can successfully generate better child solutions, and calculate new F and CR through weighted adaptive methods. The following was the formula for set parameters:

$$F_{i,j} = \text{rand}c_{i,j}(\mu F_j, 0.1) \tag{18}$$

$$\text{CR}_{i,j} = \text{rand}n_{i,j}(\mu\text{CR}_j, 0.1) \tag{19}$$

where $F_{i,j}$ is the scaling factor of individual x_i using the j-th mutation strategy, rand$c_{i,j}$ is the Cauchy distribution. If $F_{i,j} > 1$, it is truncated to 1, and if $F_{i,j} \le 0$, it is regenerated. CR$_{i,j}$ is the crossover probability of individual x_i using the jth mutation strategy, rand$n_{i,j}$ is the Normal distribution and the standard deviation is 0.1. At the end of each generation, μF_j and μCR_j are updated according to the following formula:

$$\mu F_j = (1 - c) \cdot \mu F_j + c \cdot \text{mean}_{\text{WL}}(S_{F,j}) \tag{20}$$

$$\mu\text{CR}_j = (1 - c) \cdot \mu\text{CR}_j + c \cdot \text{mean}_{\text{WL}}(S_{\text{CR},j}) \tag{21}$$

where mean$_{\text{WL}}(S_{\text{CR},j})$ and mean$_{\text{WL}}(S_{F,j})$ are the weighted mean of $S_{F,j}$ and $S_{\text{CR},j}$ respectively, and c is a normal number between 0 and 1. The mean$_{\text{WL}}(S_{\text{CR},j})$ and mean$_{\text{WL}}(S_{F,j})$

can be calculated as follows:

$$\text{mean}_{\text{WL}}(S_{\text{CR},j}) = \frac{\sum_{k=1}^{|S_{\text{CR}}|} w_k^g \cdot S_{\text{CR},k}^2}{\sum_{k=1}^{|S_{\text{CR}}|} w_k^g \cdot S_{\text{CR},k}} \tag{22}$$

$$\text{mean}_{\text{WL}}(S_{F,j}) = \frac{\sum_{k=1}^{|S_F|} w_k^g \cdot S_{F,k}^2}{\sum_{k=1}^{|S_F|} w_k^g \cdot S_{F,k}} \tag{23}$$

The weights of each F and CR were calculated by weighting the mean. The weight w_k^g is updated like that:

$$w_k = \frac{\Delta d_k}{\sum_{k=1}^{|S_{\text{CR}}| or |S_F|} \Delta d_{k,g}} \tag{24}$$

$$\Delta d_{k,g} = |f(x_{k,g}) - f(u_{k,g})| \tag{25}$$

$$w_k^{g+1} = \begin{cases} w_k^g, \, if \;\; f(x_{k,g}) < f(u_{k,g}) \\ w_k, \text{ if } (f(x_{k,g}) > f(u_{k,g}) \text{ and rand } < 0.5) \\ \delta_1 * w_k + \delta_2 * w_k^g, \text{ otherwise} \end{cases} \tag{26}$$

where $\Delta d_{k,g}$ are fitness improvements and are used to influence parameter adaptation. δ_1 and δ_2 are the weight combination coefficients.

First, if the current generation succeeded in generating a better solution, the weight continued to live into the next generation, i.e. $w_k^{g+1} = w_k^g$. Second, if the generation did not succeed in generating a better solution and the random number rand was less than 0.5, the weight was discarded and a new weight was generated through $\Delta d_{k,g}$. Third, if the contemporary generation did not succeed in generating a better solution and the random number rand was greater than 0.5, the contemporary weights were used to regenerate and calculate the weights for the next generation.

In particular, MDE-ctd applies this parameter control method to the "DE/current-to-pbest/1" mutation strategy and continues to use the parameter adaptive mechanism in JADE on other mutation strategies.

Algorithm 1 Pseudo-code of MDE-ctd.

```
1: Set $\mu F_j = 0.5$, $\mu \text{CR}_j = 0.5$ , $\Delta f_j = 0$ and $\Delta Fes_j = 0$ for each j =1,...3;
2: Initialize, $NP$, $ng$ for each $j = 1, \dots, 3$;
3: Initialize the $pop$ randomly distributed in the solution space;
4: Initialize $\lambda_j = \frac{1}{3}$ and set $NP_j = \lambda_j \cdot NP$;
5: Randomly partition pop into pop1 , pop2 , pop3 with respect to their sizes;
6: Set $gen = 0$, $Fes = 0$ and $MaxFes = D \times 10000$;
7: while $Fes \le MaxFes$ do
8:     $gen = gen + 1$
9:     if $mod(g, ng) == 0$ then
10:        $k = arg(max_{1 \le j \le 3}(\frac{\Delta f_j}{\Delta Fes_j}))$ ;
11:        $\lambda_j = \frac{score_j}{\sum_{j=1,...3} score_j}$ ;
12:        $\Delta f_j = 0$;
13:    end if
14:    Calculate the size of $NP_j$;
15:    Randomly assign $pop_1$, $pop_2$ and $pop_3$ according to $\lambda_j$ ;
16:    for $j = 1 \to 3$ do
17:        Calculate $\mu CR_j$ and $\mu F_j$;
18:        Calculate $CR_{i,j}$ and $F_{i,j}$ for each individual $x_i$ in $pop_j$;
19:        Perform the j-th mutation strategy and related crossover operators over sub-population $pop_j$;
20:        Set $S_{CR,j} = \emptyset$ and $S_{F,j} = \emptyset$;
21:    end for
22:    for $i = 1 \to NP$ do
23:        if $f(x_{i,g}) \le f(u_{i,g})$ then
24:            $x_{i,g+1} = x_{i,g}$;
25:        else
26:            $x_{i,g+1} = u_{i,g}$;
27:            $\Delta f_j = \Delta f_j + f(x_{i,g}) - f(u_{i,g})$;
28:            $CR_{i,j} \to S_{CR,j}$; $F_{i,j} \to S_{F,j}$;
29:        end if
30:    end for
31:    $Fes = Fes + NP$
32: end while
33: return the best agent fitness
```

Complexity analysis

According to the pseudo-code given in Algorithm 1, the time complexity of MDE-ctd was analyzed as follows. NP is the population size, D is the dimension of problem dimension, G_{max} is maximum number of generations. The time complexity of classical DE can be expressed as $O(G_{\max} \times \text{NP} \times D)$. As stated in [36], MPEDE only extended the JADE mutation strategy and parameter adaptability, so the total complexity of both MPEDE and JADE was $O(G_{\max} \times \text{NP} \times [D + \log(\text{NP})])$. The main difference between MDE-ctd and MPEDE was the grouping method, integrated mutation strategy and adaptive mechanism. For the grouping methods based on contribution degree and the adaptive individual parameter settings (F and CR) based on HSW, it only called the previous data in the archive, and would not increase the time complexity of the whole algorithm. Multi-population and multi-strategy integration method maintained the same complexity as in MPEDE. Therefore, considering the overall algorithm, the running complexity of MDE-ctd is $O(G_{max} \times \text{NP} \times [D + \log(\text{NP})])$.

Experimental studies

In order to fully evaluate the performance of MDE-ctd, several experiments were done on CEC 2005 global optimization benchmark function suite (30D, 50D) [37] and CEC 2014 global optimization benchmark function suite (30D) [38]. Some state-of-art algorithms (CoDE [34], JADE [16], SAKPDE [39], MPMSDE [22], EPSDE [35], SHADE [17], EDEV [21] and MPEDE [24]) are used to compare with MDE-ctd. Further, the contribution degree analysis, strategy effectiveness and parameter sensitivity are conducted to test the robustness of MDE-ctd.

Table 1 Parameter configuration of each tested algorithm

Algorithm	Parameter settings
MDE-ctd	δ_1 = 0.8, δ_2 = 0.2 , ng = 5, NP = 210, c = 0.1
JADE	p = 0.05, c = 0.1, NP = 100
CoDE	Three trial vector generation strategies and three control parameter settings, NP = 30
SAKPDE	CR range [0.3, 1.0] and F range [0.4, 1.0], NP = 100
MPMSDE	δ = 0.04, p = 0.04, ng = 25, NP = 250, c = 0.1
EPSDE	CR range [0.1, 0.9] and F range [0.4, 0.9], NP = 50
SHADE	$p_m in = \frac{2}{NP}$, H = NP = 100
EDEV	$\lambda_1 = \lambda_2 = \lambda_3$ = 0.1, λ_4 = 0.7, ng = 20, NP = 50 or NP = 100
MPEDE	$\lambda_1 = \lambda_2 = \lambda_3$ = 0.2, ng = 20, NP = 250

Benchmark functions and experimental settings

The 25 benchmark functions of the CEC2005 can be divided into the following four categories: unimodal functions (F1–F5), basic multimodal functions (F6–F12), expanded multimodal functions (F13–F14), and hybrid composition multimodal functions (F15–F25). The 30 benchmark functions of the CEC2014 can be divided into the following four categories: unimodal functions (F1–F3), simple multimodal functions (F4–F16), hybrid multimodal functions (F17–F22), and composition multimodal functions (F23–F30). The detail information on benchmark optimization functions, see [37, 38].

The parameters of MDE-ctd are set to NP = 210, ng = 5, and c = 0.1. The initial F and CR values are set to 0.5 for the archival sub-population. The initial F values of the exploratory sub-population are set to 0.5. In the integrated sub-population, CR ranged in $[0.1, 0.9]$ and F ranged in $[0.4, 0.9]$. The values of δ_1 and δ_2 in the parameter adaption based on HSW are 0.8 and 0.2, respectively. $MaxFes = 10{,}000 \times D$ is chosen as the experiment's maximum fitness evaluation number, where D is the dimension of the benchmark function. All comparison algorithms' parameter configurations (including NP, F and CR values and various additional parameters) were set to the identical guidelines on their original publications, these parameters are shown in Table 1.

Due to the algorithm's randomness, each algorithm was independently run 25 times for statistical comparisons. The mean value and standard deviation value are calculated to assess the algorithm's performance. For each problem, the best result is bolded. Results of MDE-ctd are compared with those of CoDE, JADE, SAKPDE, MPMSDE, EPSDE, SHADE, EDEV and MPEDE, respectively, by Wilcoxon rank sum test at the significance level of 0.05. The marker "$-$" is worse than the results of MDE-ctd, "$+$" is better than the results of MDE-ctd and "$\approx$" is equivalent to the results of MDE-ctd.

Comparison with advanced DE variants on CEC2005 for 30D/50D problems

From Tables 2 and 3, MDE-ctd was compared with eight DE variants in the CEC2005 benchmark function suite (30D and 50D), respectively. Several observations and conclusions can be drawn from the analysis of the experimental results. First of all, for the unimodal functions (F1–F5), at 30D, MDE-ctd was almost superior to all other algorithms, only slightly worse than SHADE on F5. In the case of 50D, except CoDE and EPSDE, other algorithms can find the global optimal solution to F1 in 50D. MDE-ctd was superior to CoDE, MPMSDE, EPSDE and MPEDE on F2, only inferior to MPMSDE, SHADE and EDEV on F3, superior to the other 5 algorithms. It was better than all other algorithms on F4–F5. For unimodal functions, MDE-ctd had the ability of fast convergence, and it can find the optimal solution quickly. This is because the "DE/current to best/1" strategy in the archival sub-population can converge quickly on the unimodal problem. More computing resources are allocated to archival sub-populations through the contribution degree, thus it can improve the overall convergence ability of the MDE-ctd.

Secondly, for the basic multimodal functions F6–F12, at 30D, MDE-ctd is inferior to SAKPDE and SHADE in F6 and superior to all other algorithms in F7. JADE, MPMSDE, EPSDE, SHADE, EDEV, MPEDE, and MDE-ctd have the same performance at F8. All the nine algorithms can find the global optimal solution on F9. MDE-ctd was superior to other algorithms except SHADE on F10, and superior to EPSDE and SHADE on F11. MDE-ctd outperformed all other algorithms on F12. In the case of 50D, MDE-ctd was superior to all other algorithms on F7, F10, F12. Only second to EDEV on F6, other algorithms except CoDE, SAKPDE and SHADE maintained the same performance on F8. MDE-ctd, JADE, MPMSDE, SHADE and EDEV can find global optimal solutions on F9, MDE-ctd was superior to JADE, EPSDE and SHADE on F11.

Table 2 Comparison results of solution accuracy on CEC2005 benchmark (30D)

Function		CoDE mean (std dev)	JADE mean (std dev)	SAKPDE mean (std dev)	MPMSDE mean (std dev)
Unimodal functions	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)**≈
	F2	5.94E−16 (3.77E−16)−	2.35E−28 (1.92E−28)−	2.03E−29 (0.00E+00)−	3.64E−29 (6.19E−29)−
	F3	9.94E+04 (2.85E+04)−	9.69E+03 (4.03E+03)−	2.43E+04 (1.77E+04)−	1.51E+02 (2.33E+02)−
	F4	7.15E−03 (1.37E−03)−	4.07E−11 (5.41E−11)−	7.21E−15 (5.82E−15)−	6.32E−20 (8.07E−19)−
	F5	2.62E+02 (3.60E+02)−	4.86E−06 (5.30E−05)−	2.49E+00 (3.86E+00)−	4.64E−06 (7.91E−06)−
Basic multimodal functions	F6	5.10E−01 (7.20E−02)−	1.61E+01 (3.25E+01)−	**0.00E+00 (0.00E+00)+**	3.66E−01 (1.56E+00)−
	F7	7.53E−03 (8.55E−03)−	4.70E+03 (1.77E−12)−	4.89E−03 (7.54E−03)−	3.99E−03 (6.73E−03)−
	F8	**2.01E+01 (3.55E−02)+**	2.09E+01 (4.78E−02)≈	**2.01E+01 (2.87E−01)+**	2.09E+01 (5.49E−02)≈
	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)**≈
	F10	3.58E+01 (2.81E+00)−	2.49E+01 (5.04E+00)−	3.56E+01 (2.14E+00)−	2.42E+01 (7.02E+00)−
	F11	**1.56E+01 (6.42E+00)+**	2.48E+01 (5.00E−01)+	2.35E+01 (5.76E+00)+	1.60E+01 (6.28E+00)+
	F12	6.52E+03 (1.88E+03)−	5.71E+03 (4.90E+03)−	2.91E+03 (2.45E+03)−	2.21E+03 (2.85E+03)−
Expanded multimodal functions	F13	1.64E+00 (1.03E−01)+	1.44E+00 (5.75E−02)+	2.21E+00 (6.42E−01)−	1.94E+00 (2.58E−01)≈
	F14	**1.23E+01 (4.91E−02)**≈	**1.23E+01 (8.32E−02)**≈	**1.23E+01 (5.37E−01)**≈	**1.22E+01 (3.06E−01)**≈
Hybrid composition functions	F15	4.54E+02 (7.07E+01)−	4.33E+02 (5.77E+01)−	3.54E+02 (6.69E+01)−	3.69E+02 (2.55E+02)−
	F16	7.23E+01 (1.86E+01)−	4.79E+01 (8.54E+00)−	6.31E+01 (1.08E+01)−	5.51E+01 (1.97E+01)−
	F17	7.68E+01 (1.84E+00)−	7.18E+01 (1.84E+01)−	5.56E+01 (1.28E+01)−	5.72E+01 (2.35E+01)−
	F18	9.05E+02 (1.79E+00)−	9.04E+02 (5.38E−01)−	9.00E+02 (0.00E+00)−	9.04E+02 (4.87E−01)−
	F19	9.04E+02 (1.07E−01)−	9.06E+02 (1.23E+00)−	9.00E+02 (0.00E+00)−	9.04E+02 (5.59E−01)−
	F20	9.04E+02 (4.66E−02)−	9.04E+02 (2.36E−01)−	9.00E+02 (0.00E+00)−	9.04E+02 (4.07E−01)−
	F21	**5.00E+02 (8.08E**−06)≈	**5.00E+02 (2.38E−04)**≈	**5.00E+02 (2.87E−13)**≈	**5.00E+02 (2.18E−13)**≈
	F22	8.59E+02 (3.26E+01)−	8.62E+02 (1.50E+01)−	8.62E+02 (1.89E+01)−	8.63E+02 (2.18E+01)−
	F23	**5.34E+02 (3.98E−04)**≈	**5.34E+02 (1.14E−03)**≈	**5.34E+02 (5.61E−04)**≈	**5.34E+02 (7.83E−07)**≈
	F24	**2.00E+02 (2.56E−14)+**	**2.00E+02 (2.21E−14)+**	**2.00E+02 (3.18E−14)+**	**2.00E+02 (3.39E−14)+**
	F25	**2.10E+02 (5.35E−01)**≈	1.63E+03 (2.85E+00)−	**2.10E+02 (7.87E−01)**≈	**2.10E+02 (6.19E−01)**≈
-/+/≈		15/4/6	16/3/6	15/4/6	15/2/8

Table 2 continued

Function		EPSDE mean (std dev)	SHADE mean (std dev)	EDEV mean (std dev)	MPEDE mean (std dev)	MDE-ctd mean (std dev)
Unimodal functions	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)**
	F2	3.28E−14 (1.59E−15)−	6.34E−29 (7.62E−29)−	3.71E−28 (8.50E−29)−	1.88E−27 (1.59E−27)−	**7.68E−30 (1.24E−29)**
	F3	8.47E+05 (5.41E+06)−	8.04E+03 (4.86E+03)−	1.28E+04 (7.69E+03)−	2.23E+01 (1.60E+02)−	**7.12E−02 (5.99E−02)**
	F4	2.38E+02 (4.76E+02)−	4.35E−16 (4.77E−16)−	9.37E−09 (1.62E−08)−	8.90E−18 (1.01E−17)−	**5.44E−23 (3.95E−23)**
	F5	1.41E+03 (8.51E+02)−	**1.46E−10 (3.32E−10)+**	2.07E−02 (2.47E−02)−	3.70E−06 (6.71E−06)−	6.30E−07 (1.24E−07)
Basic multimodal functions	F6	3.89E−01 (1.35E+00)−	1.28E−27 (1.60E−27)+	4.66E−01 (2.87E−01)−	1.45E+00 (2.52E+00)−	6.02E−16 (4.38E−16)
	F7	1.06E+00 (3.92E−01)−	7.40E−03 (1.69E−04)−	9.37E−03 (5.83E−03)−	3.29E−03 (5.70E−03)−	**1.23E−03 (3.89E−03)**
	F8	2.09E+01 (3.56E−02)≈	2.07E+01 (2.72E−02)≈	2.07E+01 (5.24E−02)≈	2.10E+01 (3.82E−02)≈	2.09E+01 (3.74E−02)
	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)**
	F10	9.78E+01 (1.83E+01)−	**1.24E+01 (2.51E−01)+**	2.47E+01 (5.61E+00)−	2.23E+01 (1.09E+01)−	1.77E+01 (4.97E+00)
	F11	3.54E+01 (4.29E+00)−	2.82E+01 (1.01E+00)−	2.16E+01 (8.18E+00)+	2.53E+01 (1.10E+00)+	2.56E+01 (4.41E−01)
	F12	3.86E+04 (5.19E+03)−	3.06E+03 (1.28E+03)−	2.29E+03 (4.64E+03)−	1.54E+03 (9.24E+02)−	**1.29E+03 (7.17E+02)**
Expanded multimodal functions	F13	2.03E+00 (2.05E−01)−	**1.08E+00 (7.30E−02)+**	1.45E+00 (5.75E−01)+	2.44E+00 (4.33E−01)−	1.93E+00 (2.55E−01)
	F14	1.34E+01 (3.78E−01)−	1.26E+01 (4.09E−03)−	**1.22E+01 (1.34E−01)**≈	**1.24E+01 (1.05E−01)**≈	**1.23E+01 (4.79E−01)**
Hybrid composition functions	F15	**2.17E+02 (4.13E+01)+**	4.51E+02 (7.07E+01)−	3.82E+02 (3.12E+01)−	3.77E+02 (1.96E+02)−	3.14E+02 (9.91E+01)
	F16	1.39E+02 (7.58E+01)−	1.31E+02 (1.27E+02)−	8.77E+01 (1.94E+02)−	6.62E+01 (2.08E+01)−	**4.39E+01 (5.89E+00)**
	F17	1.82E+02 (2.31E+02)−	1.09E+02 (6.84E+01)−	1.05E+02 (5.19E+01)−	**3.76E+01 (6.06E+00)+**	4.18E+01 (6.27E+00)
	F18	8.23E+02 (5.17E+00)−	9.04E+02 (2.34E−01)−	8.16E+02 (1.58E+00)≈	9.04E+02 (2.65E−01)−	**8.16E+02 (2.12E−01)**
	F19	8.26E+02 (4.18E+00)−	9.04E+02 (1.97E−01)−	8.22E+02 (3.63E+00)−	9.04E+02 (1.89E−01)−	**8.17E+02 (3.96E−01)**
	F20	8.23E+02 (4.03E+00)−	9.04E+02 (2.01E−01)−	8.27E+02 (2.82E+00)−	9.04E+02 (2.85E−01)−	**8.17E+02 (7.07E−02)**
	F21	8.35E+02 (6.71E+01)−	**5.00E+02 (8.36E−14)**≈	**5.00E+02 (3.17E−13)**≈	**5.00E+02 (3.59E−05)**≈	5.00E+02 (4.62E−06)
	F22	5.06E+02 (4.54E+00)−	8.49E+02 (1.93E+01)−	5.23E+02 (2.27E+01)−	8.73E+02 (2.18E+01)−	**5.02E+02 (7.07E+00)**
	F23	8.29E+02 (5.78E+01)−	**5.34E+02 (5.68E−13)**≈	**5.34E+02 (9.24E−06)**≈	**5.34E+02 (4.97E−04)**≈	**5.34E+02 (5.82E−05)**
	F24	2.11E+02 (1.92E−13)≈	**2.00E+02 (8.59E−13)+**	**2.00E+02 (5.33E−13)+**	**2.00E+02 (2.32E−14)+**	2.10E+02 (2.17E−14)
	F25	2.13E+02 (4.56E+00)−	**2.10E+02 (4.38E−01)**≈	2.12E+02 (1.05E+00)−	**2.09E+02 (6.76E−01)**≈	**2.09E+02 (1.13E−01)**
-/+/≈		20/1/4	14/5/6	15/3/7	15/3/7	

Table 3 Comparison results of solution accuracy on CEC2005 benchmark (50D)

Function		CoDE mean (std dev)	JADE mean (std dev)	SAKPDE mean (std dev)	MPMSDE mean (std dev)
Unimodal functions	F1	2.52E−29 (3.57E−29)−	**0.00E+00 (0.00E+00)**≈	**0.00E+00 (0.00E+00)**≈	**0.00E+00 (0.00E+00)**≈
	F2	1.83E−09 (8.40E−10)−	**4.53E−27 (2.14E−27)+**	2.42E−25 (8.61E−24)+	4.58E−16 (3.97E−15)−
	F3	1.05E+05 (2.02E+04)−	1.19E+05 (5.39E+04)−	2.28E+05 (8.76E+04)−	5.70E+04 (2.81E+04)+
	F4	7.25E+02 (8.23E+02)−	1.18E+00 (7.59E−01)−	1.35E+01 (7.87E+01)−	2.39E−01 (4.16E−01)−
	F5	4.28E+03 (7.00E+02)−	1.63E+03 (2.15E+02)−	3.05E+03 (5.92E+02)−	7.11E+02 (3.63E+02)−
Basic multimodal functions	F6	4.13E+01 (5.62E+01)−	6.95E+00 (1.20E+01)−	2.05E−01 (8.31E−01)−	3.47E−01 (1.04E+00)−
	F7	2.18E−03 (3.62E−03)−	6.20E−03 (4.91E−04)−	2.76E−02 (8.29E−02)−	7.36E−03 (2.51E−02)−
	F8	**2.01E+01 (1.73E−01)+**	2.11E+01 (8.51E−02)≈	**2.01E+01 (2.07E−01)+**	2.11E+01 (2.87E−02)≈
	F9	6.18E−01 (5.33E−01)−	**0.00E+00 (0.00E+00)**≈	6.87E−01 (9.59E−01)−	**0.00E+00 (0.00E+00)**≈
	F10	7.76E+01 (8.44E+00)−	4.69E+01 (7.94E+00)−	8.29E+01 (3.62E+01)−	4.55E+01 (1.43E+01)−
	F11	3.38E+01 (3.71E+00)+	5.05E+01 (4.44E+00)−	3.27E+01 (6.13E+00)+	**8.74E−02 (1.29E−01)+**
	F12	1.65E+04 (1.47E+03)−	1.78E+04 (1.39E+04)−	2.67E+04 (1.92E+04)−	8.29E+03 (5.73E+03)−
Expanded multimodal functions	F13	3.55E+00 (2.86E−02)+	2.73E+00 (7.40E−02)+	4.86E+00 (9.41E−01)−	4.47E+00 (2.54E+00)−
	F14	2.21E+01 (1.68E−01)−	**2.18E+01 (3.93E−01)**≈	2.21E+01 (4.78E−01)−	**2.18E+01 (5.94E−01)**≈
Hybrid composition functions	F15	3.94E+02 (8.25E+00)−	3.67E+02 (5.77E+01)−	2.32E+02 (8.07E+01)+	3.56E+02 (7.83E+01)−
	F16	8.57E+01 (3.46E+00)−	4.41E+01 (3.77E+00)−	6.08E+01 (2.62E+01)−	7.33E+01 (6.92E+01)−
	F17	5.64E+01 (8.29E+00)+	1.03E+02 (1.63E+01)−	7.72E+01 (3.62E+01)−	6.02E+01 (7.88E+01)+
	F18	9.24E+02 (2.65E+00)−	9.15E+02 (1.90E+01)−	9.05E+02 (2.03E+00)−	9.19E+02 (6.55E+00)−
	F19	9.21E+02 (9.64E−01)−	9.34E+02 (2.12E+01)−	9.04E+02 (3.26E+01)−	9.17E+02 (5.68E+00)−
	F20	9.26E+02 (9.69E+00)−	9.20E+02 (1.25E+00)−	9.02E+02 (1.53E+00)−	9.19E+02 (2.85E+00)−
	F21	7.55E+02 (3.60E+02)−	5.22E+02 (2.93E+01)+	**5.00E+02 (6.24E−13)+**	7.93E+02 (2.65E+02)−
	F22	9.04E+02 (3.10E+00)−	9.19E+02 (2.25E+01)−	8.86E+02 (3.12E+01)−	9.22E+02 (2.35E+01)−
	F23	7.25E+02 (7.07E+00)≈	**5.34E+02 (2.26-05)+**	5.37E+02 (1.70E−01)+	9.64E+02 (2.17E+02)−
	F24	**2.00E+02 (2.42E−09)+**	**2.00E+02 (1.74E−12)+**	**2.00E+02 (8.62E−11)+**	**2.00E+02 (8.27E−12)+**
	F25	2.18E+02 (4.99E−01)−	1.67E+03 (3.68E+00)−	**2.16E+02 (1.42E+00)**≈	**2.16E+02 (2.08E+00)**≈
-/+/≈		19/5/1	16/5/4	16/7/2	16/4/5

Table 3 continued

Function		CoDE mean (std dev)	JADE mean (std dev)	SAKPDE mean (std dev)	MPMSDE mean (std dev)	
Unimodal functions	F1	3.94E−29(1.85E−29)−	**0.00E+00 (0.00E+00)**≈	**0.00E+00 (0.00E+00)**≈	**0.00E+00 (0.00E+00)**≈	**0.00E+00 (0.00E+00)**
	F2	5.85E+02 (8.27E+02)−	1.50E−26 (9.43E−27)+	1.90E−20 (1.26E−20)+	4.44E−13 (5.03E−13)−	8.29E−17 (1.01E−16)
	F3	4.74E+06 (4.36E+06)−	**2.37E+04 (9.52E+03)+**	5.50E+04 (2.32E+04)+	6.05E+04 (1.19E+04)−	5.91E+04 (8.10E+03)
	F4	4.22E+03 (1.13E+03)−	1.58E+00 (1.66E+00)−	9.41E−01 (8.43E−01)−	3.47E+00 (5.77E+00)−	**3.10E−02 (3.72E−02)**
	F5	4.57E+03 (1.27E+02)−	1.19E+03 (2.48E+02)−	1.50E+03 (8.32E+02)−	3.48E+02 (3.49E+02)−	**2.92E+02 (3.16E+02)**
Basic multimodal functions	F6	7.19E+01 (1.01E+02)−	4.92E−01 (3.54E−02)−	**1.12E−24 (1.25E−24)+**	2.66E+00 (2.30E+00)−	4.15E−05 (7.57E−05)
	F7	5.25E−01 (7.28E−01)−	7.39E−03 (1.04E−02)−	7.27E−03 (2.58E−03)−	2.54E−03 (4.21E−03)−	**1.73E−03 (3.21E−03)**
	F8	2.11E+01 (3.16E−02)≈	2.07E+01 (1.28E−01)+	2.11E+01 (1.15E−02)≈	2.11E+01 (1.30E−02)≈	2.11E+01 (5.88E−02)
	F9	6.04E−16 (5.02E−17)−	**0.00E+00 (0.00E+00)**≈	**0.00E+00 (0.00E+00)**≈	2.78E−07 (4.82E−07)−	**0.00E+00 (0.00E+00)**
	F10	1.55E+02 (7.07E−01)−	4.17E+01 (5.22E+00)−	5.17E+02 (2.85E+02)−	4.71E+01 (3.49E+00)−	**3.95E+01 (1.64E+01)**
	F11	6.96E+01 (1.20E+00)−	5.30E+01 (3.24E+00)−	4.34E+02 (9.88E+01)−	3.22E+01 (1.39E+01)+	4.77E+01 (1.09E+01)
	F12	3.22E+05 (8.49E+03)−	6.25E+04 (7.86E+03)−	4.81E+04 (6.39E+03)−	9.05E+03 (1.10E+04)−	**6.95E+03 (6.13E+03)**
Expanded multimodal functions	F13	6.12E+00 (7.78E−02)−	**2.10E+00 (2.06E−02)+**	3.97E+00 (3.66E−01)−	3.61E+00 (5.90E−01)+	3.70E+00 (2.04E−01)
	F14	2.34E+01 (0.00E+00)−	**2.18E+01 (4.25E−01)**≈	**2.18E+01 (5.94E−01)**≈	**2.18E+01 (2.76E−02)**≈	**2.16E+01 (4.14E−01)**
Hybrid composition functions	F15	**2.62E+02 (3.54E+00)+**	3.22E+02 (6.54E−01)−	3.67E+02 (5.77E+01)−	3.37E+02 (5.47E+01)−	3.02E+02 (1.08E+02)
	F16	1.47E+02 (4.24E+00)−	3.64E+01 (7.87E+00)−	6.15E+01 (2.24E+01)−	3.78E+01 (5.18E+00)−	**3.47E+01 (8.04E+00)**
	F17	2.21E+02 (2.47E+01)−	7.66E+01 (7.56E+00)−	8.16E+01 (3.87E+01)−	**4.01E+01 (6.25E+00)+**	7.06E+01 (4.54E+01)
	F18	**8.50E+02 (4.95E+00)+**	9.19E+02 (4.61E+00)−	8.77E+02 (2.71E+01)+	9.20E+02 (1.23E+00)−	8.82E+02 (4.27E+01)
	F19	**8.53E+02 (9.19E+00)+**	9.19E+02 (5.96E+00)−	8.84E+02 (4.56E+01)−	9.18E+02 (2.66E+00)−	8.77E+02 (4.37E+01)
	F20	**8.51E+02 (7.07E+00)+**	9.16E+02 (4.89E+00)−	8.97E+02 (1.68E+01)−	9.19E+02 (6.44E−01)−	8.71E+02 (4.15E+01)
	F21	7.30E+02 (1.41E+00)−	5.09E+02 (9.19E+00)+	5.76E+02 (1.31E+02)+	**5.00E+02 (3.11E−04)+**	7.20E+02 (3.07E+00)
	F22	**5.00E+02 (6.92E−02)**≈	9.08E+02 (2.75E+01)−	5.03E+02 (5.55E−01)−	8.89E+02 (1.10E+01)−	**5.00E+02 (4.34E−02)**
	F23	7.31E+02 (2.83E+00)−	5.39E+02 (4.42E−02)+	5.39E+02 (3.46E−02)+	**5.34E+02 (4.04E−02)+**	7.24E+02 (1.43E+00)
	F24	2.39E+02 (1.41E+00)−	**2.00E+02 (2.67E−13)+**	**2.00E+02 (3.24E−12)+**	**2.00E+02 (1.31E−12)+**	2.20E+02 (2.71E+00)
	F25	2.45E+02 (2.83E+00)−	2.18E+02 (2.20E+00)−	**2.16E+02 (4.61E−01)**≈	**2.15E+02 (8.30E−01)**≈	**2.15E+02 (1.14E+00)**
-/+/≈		19/4/2	15/7/3	13/7/5	15/6/4	

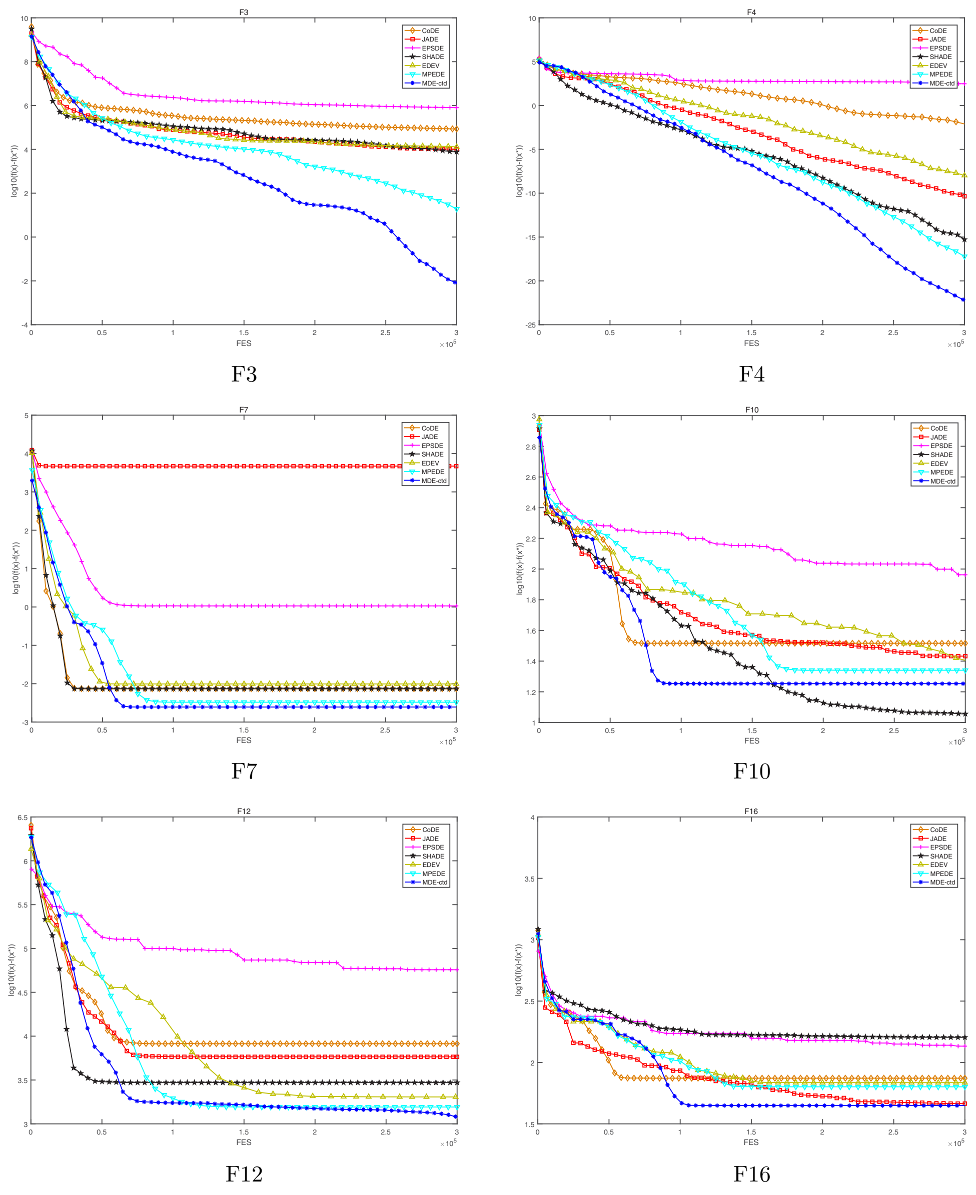


Fig. 2 Convergence curves of 9 algorithms on F3, F4, F7, F10, F12, F16 representative test functions of 30D

Thirdly, for the expanded multimodal functions F13–F14, SHADE performed best on F13. In the 30D case of F14, CoDE, JADE, SAKPDE, MPMSDE, EDEV, MPEDE, and MDE-ctd maintained the same performance, and in the 50D case, JADE, MPMSDE, SHADE, EDEV, MPEDE and MDE-ctd have the same performance. For multimodal functions, MDE-ctd can balance exploration and development, and maintain population diversity through multi-population and multi-strategy integration method.

Finally, for the hybrid composition multimodal functions F15–F25, at 30D, MDE-ctd was second only to EPSDE and MPEDE on F15 and F17 respectively, and MDE-ctd maintained an advantage on F16, F18–F23 and F25. At 50D, MDE-ctd is second only to SAKPDE and EPSDE in F15, second only to EPSDE in F18–F20, superior to all other algorithms on F16, and second only to MPMSDE and MPEDE in F17. SAKPDE and MPEDE were superior to all other algorithms on F21, MDE-ctd and EPSDE are superior to the other 7 algorithms on F22, MED-ctd, SAKPDE, MPMSDE, EDEV and MPEDE were superior to the other 4 algorithms on F25. For hybrid composition multimodal functions, MDE-ctd can not only ensure that the optimal mutation strategy obtains enough computing resources, but also use the multi-population and multi-strategy integration method to search more space and improve the diversity of the algorithm, so as to jump out of the local optimum.

In addition, the convergence rate was also a key indicator of these algorithms, the convergence curves of portion comparison algorithms on some selected functions of 30D were plotted in Figs. 2 and 3 to observe their evolution. The

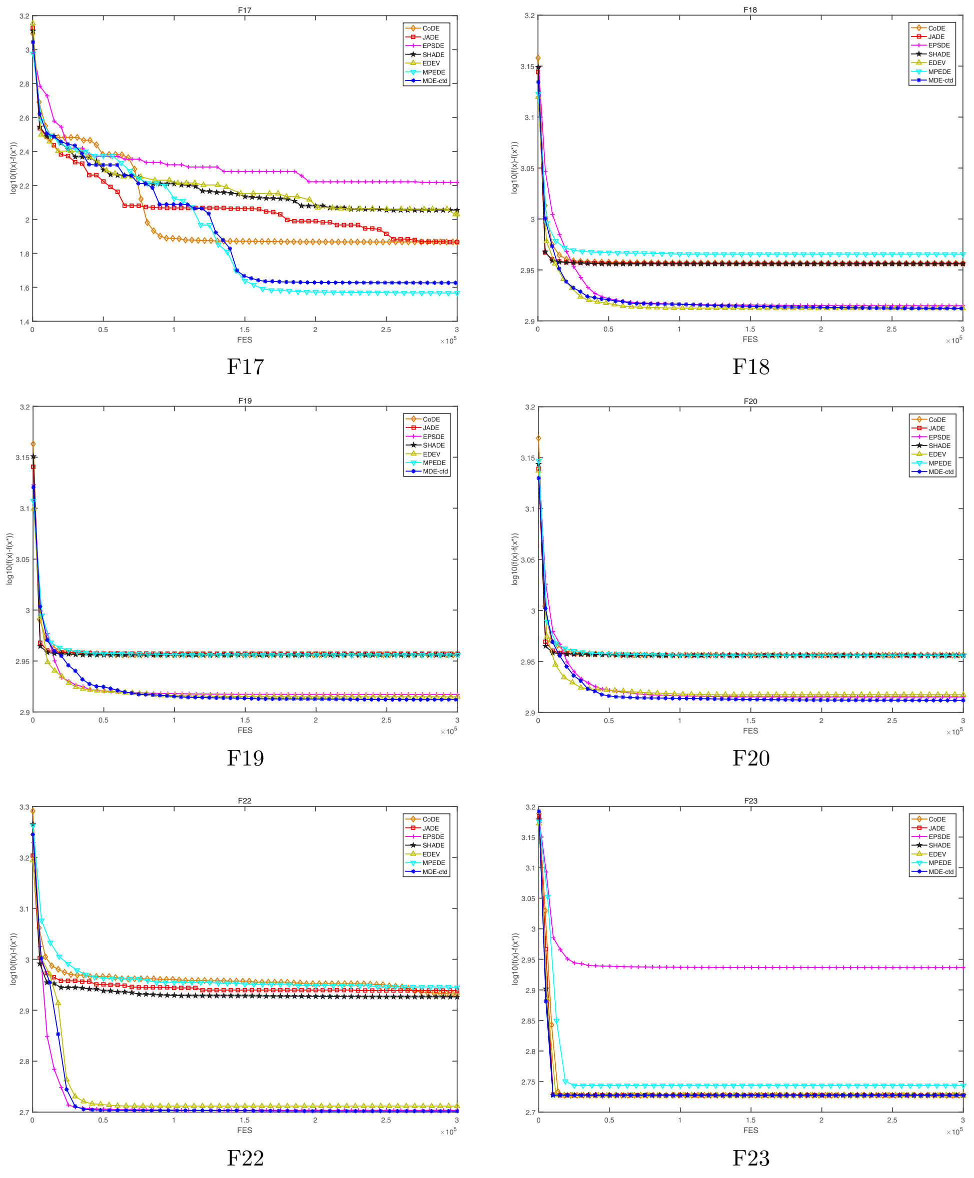


Fig. 3 Convergence curves of 9 algorithms on F17, F18, F19, F20, F22, F23 representative test functions of 30D

performance index was measured by the mean function error $(f(x) - f(x^*))$, where x was the contemporary optimal solution to evolution and x^* was the global optimization of the best function. As seen in Figs. 2 and 3, MDE-ctd can converge quickly and is superior to other DE algorithms on Unimodal functions F3 and F4. On the basic multimodal functions F7 and F12, it can be observed that MDE-ctd can get better solutions than other DE algorithms. On F10, MDE-ctd had a faster convergence rate in the early stage of evolution. On the Hybrid composition functions F16–F20 and F22–F23, the convergence rate of MDE-ctd at F16, F18, F19 and F22 were a little slower in the early stage of evolution, but it can get better solutions than other DE algorithms in the late stage of evolution. On F17, MPEDE and MDE-ctd obtained the best solution. On F23, EPSDE converged slowly, while other DE algorithms can achieve a similar convergence speed.

In conclusion, MDE-ctd has obvious advantages over several other state-of-art DE algorithms. The reason that MDE-ctd is superior to other algorithms is that MDE-ctd can evolve simultaneously in the mutual influence using multi-mutation strategies. These mutation strategies can work independently, enabling MDE-ctd to solve different types of optimization problems. In addition, the contribution degree grouping method used in MDE-ctd allows more computing resources to be allocated to different sub-populations more reasonably, which is faster than other multi-population or multi-strategy algorithms to obtain feedback in the evolution process and locate the optimal solution quickly.

Comparison with advanced DE variants on CEC2014 for 30D problems

MDE-ctd was compared with eight state-of-art DE variants on CEC 2014 (30 benchmark functions) to test its performance. The comparison of MDE-ctd and eight DE variants was summarized in Table 4. The following conclusions were drawn from comparing and analyzing the experiments.

On the unimodal functioned F1–F3, MDE-ctd performs better than other algorithms. On F2, MDE-ctd, CoDE, SAKPDE, MPMSDE, EDEV, and MPEDE can all locate the global optimal solution. On F3, only SAKPDE, EDEV and MDE-ctd can do so.

On the simple multimodal functions F4–F16, MDE-ctd outperformed other algorithms on F4 and F13. On F5, the 7 algorithms have the same performance. On F6 and F14, MDE-ctd was second only to SHADE and EDEV respectively. On F7, MDE-ctd, SAKPDE, MPMSDE, CoDE, EDEV and MPEDE can find globally optimal solutions. On F9, it was only surpassed by JADE and SHADE, while on F12, MDE-ctd outperformed SAKPDE, MPMSDE, EPSDE, EDEV, and MPEDE. On F15, MDE-ctd has the same performance as JADE and outperforms MPMSDE, EPSDE, EDDV, and MPEDE.

In the mixed multimodal functions F17–F22, MDE-ctd was second only to MPEDE on F17 and F18 and superior to JADE, EPSDE, EDEV and MPEDE on F19. On F20-F22, MDE-ctd outperforms all other algorithms. On the Compound multimodal function F23–F30, nine algorithms have the same performance on F23. On F24, F25 and F26, MDE-ctd achieved a comparable level of performance with 6 algorithms, respectively. On F27–F30, only EPSDE on F29 have a similar performance with MDE-ctd. In other cases, MDE-ctd is superior to other algorithms.

These results show that MDE-ctd has good performance on the CEC2014 benchmark function (unimodal functioned F1–F3, hybrid functions F17–F22 and composition functions F23–F30). Indeed, for unimodal functions, the contribution grouping method used by MDE-ctd can ensure that the best mutation strategy obtains more computing resources. For hybrid functions composition functions, multi-population and multi-strategy integration methods can maintain good diversity of the population and increase the ability of the algorithm to jump out of the local optimum.

Contribution degree analysis

In order to study the contribution degree of the three sub-populations (archival sub-population, exploratory sub-population, integrated sub-population) of MDE-ctd in the whole evolutionary process, some comparative experiments were tested on CEC2005 (30D and 50D) benchmark functions suite respectively. The experimental results are summarized in Table 5 and Fig. 4. Due to the randomness of the algorithm, the MDE-ctd was run independently 25 times. The contribution degree of each sub-population in the evolution process was evaluated by calculating the proportion of the average contribution degree to the total. For clarity, the sub-populations with the highest contribution were marked in boldface.

In different benchmark functions and different stages of evolution, the contribution degree of the three sub-populations is completely different. The contribution of the archival sub-population on F1–F9, F11–F15, and F25 was larger than the exploratory sub-population and the integrated sub-population on 30D. On F10 and F21, the exploratory sub-population contributed more than the other two sub-populations. The contribution of the archival subspecies and the integrated subspecies on F16–F17 is similar. The contribution of Integrated sub-population is always superior to archival sub-population and exploratory sub-population on F18–F20. tion of 50D, the contribution of archival sub-population was higher than the exploratory sub-population and integrated sub-population on F1–F7, F9–F17, F19–F20 and F25. Among the remaining functions, the contribution degree of the integrated sub-population was always better than archival sub-population and exploratory sub-population. Therefore, in the evolution process, the contribution degree can reflect the interaction between the constituent strategies to a certain extent, and allocate computing resources according to the contributions to different sub-populations, which is conducive to the full use of computing resources.

Strategy effectiveness

In order to verify the effectiveness of the method proposed in this Manuscript. MDE-ctd (c1), MDE-ctd (c2) and standard MDE-ctd were compared in the CEC2005 function. MDE-ctd (c1) is MDE-ctd without grouping methods based on contribution degree, while MDE-ctd(c2) is MDE-ctd without multi-population and multi-strategy integration method. Each algorithm is run independently 25 times and calculate the mean value and standard deviation to evaluate the performance of the algorithm. Among them, The experimental results are shown in Table 6, Figs. 5 and 6.

The experimental results show that MDE-ctd (c1) performs poorly in unimodal function (F1–F5), basic multimodal function (F6–F12), and extended multimodal function (F13–F14). It is not only the rate of convergence is slow, but also the optimal solution cannot be localized.

This is because MDE-ctd(c1) uses the equal grouping method to allocate the same computing resources to each sub-population. For different optimization problems, the equal grouping method can not effectively use because the feedback information to adjust the population size in time, and it can not make the entire population close to because the opti-

Table 4 Comparison results of solution accuracy on CEC2014 benchmark (30D)

Function		CoDE mean (std dev)	JADE mean (std dev)	SAKPDE mean (std dev)	MPMSDE mean (std dev)
Unimodal functions	F1	3.05E+04 (1.05E+04)−	2.72E+03 (3.08E+03)−	5.64E+03 (4.85E+04)−	2.76E−12 (3.48E−13)−
	F2	**0.00E+00 (0.00E+00)**≈	2.84E−14 (0.00E+00)−	**0.00E+00 (0.00E+00)**≈	**0.00E+00 (0.00E+00)**≈
	F3	3.79E−14 (4.28E−14)−	1.37E−11 (2.37E−11)−	**0.00E+00 (0.00E+00)**≈	1.82E−14 (2.71E−14)−
Multimodal functions	F4	7.20E−01 (6.35E−01)−	1.33E−13 (3.28E−14)−	6.34E−01 (8.12E+00)−	4.33E−11 (3.27E−10)−
	F5	**2.00E+01 (8.38E−03)+**	2.03E+01 (4.83E−03)≈	**2.01E+01 (5.67E−03)+**	2.04E+01 (3.63E−02)≈
	F6	2.17E+00 (9.40E−01)−	1.09E+01 (8.23E−01)−	2.74E+00 (9.21E−01)−	2.41E+00 (8.79E−01)−
	F7	**0.00E+00 (0.00E+00)**≈	2.03E−04 (1.76E−04)−	**0.00E+00 (0.00E+00)**≈	**0.00E+00 (0.00E+00)**≈
	F8	**0.00E+00 (0.00E+00)+**	**0.00E+00 (0.00E+00)+**	**0.00E+00 (0.00E+00)+**	**0.00E+00 (0.00E+00)+**
	F9	3.63E+01 (4.91E+00)−	2.42E+01 (1.97E+00)+	5.13E+01 (3.35E+00)−	3.64E+01 (6.70E+00)−
	F10	5.50E−01 (5.08E−01)+	7.75E−03 (8.35E−03)+	1.57E+01 (1.79E+00)−	9.77E+00 (4.19E+00)−
	F11	1.75E+03 (2.19E+01)+	1.76E+03 (2.12E+02)+	3.04E+03 (5.28E+02)−	2.83E+03 (3.59E+02)−
	F12	**8.65E−02 (1.61E−03)+**	2.78E−01 (3.19E−02)+	9.31E−01 (2.74E−01)−	6.76E−01 (3.85E−01)−
	F13	2.31E−01 (5.31E−02)−	1.98E−01 (1.15E−02)−	1.93E−01 (7.83E−02)−	3.46E−01 (3.06E−02)−
	F14	2.26E−01 (5.22E−02)−	2.44E−01 (4.15E−02)−	3.91E−01 (6.67E−02)−	2.49E−01 (3.23E−02)−
	F15	**2.73E+00 (7.32E−01)+**	3.46E+00 (2.62E−01)≈	3.32E+00 (1.02E+00)+	4.07E+00 (5.93E−01)−
	F16	9.68E+00 (5.26E−01)+	**9.17E+00 (6.04E−01)+**	1.06E+01 (6.43E+00)≈	1.12E+01 (6.21E−01)−
Hybrid functions	F17	1.15E+03 (2.25E+02)−	9.67E+02 (1.85E+01)−	4.43E+02 (3.27E+01)−	3.82E+02 (3.46E+02)−
	F18	1.84E+01 (1.44E+00)+	2.27E+02 (2.07E+02)−	3.08E+01 (6.72E+01)−	1.31E+01 (3.54E+00)−
	F19	**2.77E+00 (1.71E−01)+**	4.55E+00 (3.18E−01)−	3.46E+00 (8.18E−01)+	4.12E+00 (7.18E−01)+
	F20	1.11E+01 (2.65E+00)−	2.11E+03 (2.76E+03)−	2.71E+01 (9.79E+00)−	**7.88E+00 (6.09E +00)**≈
	F21	2.26E+02 (1.08E+02)−	2.86E+02 (1.12E+02)−	3.31E+02 (4.83E+02)−	2.44E+02 (6.80E+01)−
	F22	2.21E+02 (4.68E−03)−	1.31E+02 (6.66E+01)−	4.54E+02 (1.19E+02)−	5.11E+02 (4.36E+02)−
Composition functions	F23	**3.15E+02 (4.22E−13)**≈	**3.15E+02 (1.32E−14)**≈	**3.15E+02 (2.74E−14)**≈	**3.16E+02 (4.39E−14)**≈
	F24	**2.25E+02 (1.40E+00)**≈	**2.24E+02 (9.54E−01)**≈	**2.25E+02 (5.35E−01)**≈	**2.24E+02 (4.32E+00)**≈
	F25	**2.00E+02 (1.60E−02)**≈	2.07E+02 (3.50E+00)−	**2.00E+02 (2.37E−01)**≈	**2.02E+02 (2.37E−01)**≈
	F26	**1.00E+02 (8.79E−02)**≈	**1.00E+02 (1.14E−01)**≈	**1.00E+02 (1.79E−01)**≈	**1.00E+02 (2.74E−02)**≈
	F27	4.01E+02 (3.72E−01)−	3.72E+02 (6.20E+01)−	4.75E+02 (4.34E+01)−	3.83E+02 (7.66E+01)−
	F28	7.97E+02 (8.50E−01)−	8.10E+02 (1.76E+01)−	7.62E+02 (3.88E+00)−	8.07E+02 (9.40E+00)−
	F29	8.74E+02 (1.02E+02)−	7.30E+02 (1.20E+01)−	3.75E+02 (2.73E+01)−	5.12E+03 (4.68E+03)−
	F30	8.68E+02 (1.83E+02)−	1.45E+03 (3.50E+02)−	8.90E+02 (3.42E+02)−	7.73E+02 (3.56E+02)−
-/+/≈		15/9/6	19/6/5	18/4/8	20/2/8

Table 4 continued

Function		EPSDE Mean (std dev)	SHADE mean (std dev)	EDEV mean (std dev)	MPEDE mean (std dev)	MDE-ctd mean (std dev)
Unimodal functions	F1	1.33E+04 (2.83E+03)−	8.52E+02 (1.20E+01)−	4.35E+02 (1.27E+02)−	4.55E−03 (1.63E−02)−	**1.42E−14 (1.76E−15)**
	F2	5.04E−14 (2.12E−14)−	7.13E−15 (7.48E−15)−	**0.00E+00 (0.00E+00)** ≈	**0.00E+00 (0.00E+00)**≈	**0.00E+00 (0.00E+00)**
	F3	2.95E−12 (1.06E−13)−	5.68E−14 (3.57E−14)−	**0.00E+00 (0.00E+00)**≈	3.79E−14 (3.28E−14)−	**0.00E+00 (0.00E+00)**
Multimodal functions	F4	3.71E+00 (1.13E−01)−	8.53E−14 (4.02E−14)−	5.77E−09 (4.21E−08)−	6.32E+00 (1.10E−01)−	**3.79E−14 (4.64E−14)**
	F5	2.05E+01 (7.07E−02)≈	2.03E+01 (1.17E−02)≈	2.04E+01 (3.95E−02)≈	2.04E+01 (3.79E−02)≈	2.04E+01 (7.74E−02)
	F6	1.92E+01 (1.41E+00)−	**5.44E−01 (5.64E−01)+**	1.63E+00 (1.07E+00)−	1.33E+01 (5.46E−01)−	8.87E−01 (5.16E−01)
	F7	5.97E−04 (2.05E−05)−	2.71E−14 (2.34E−14)−	**0.00E+00 (0.00E+00)**≈	**0.00E+00 (0.00E+00)**≈	**0.00E+00 (0.00E+00)**
	F8	**0.00E+00 (0.00E+00)+**	**0.00E+00 (0.00E+00)+**	**0.00E+00 (0.00E+00)+**	3.98E−14 (6.41E−14)+	9.33E−07 (1.50E−06)
	F9	3.92E+01 (7.07E+00)−	**1.76E+01 (3.34E+00)+**	3.42E+01 (1.03E+01)−	2.75E+01 (3.76E+00)−	2.57E+01 (5.11E+00)
	F10	7.35E−02 (7.86E−02)+	**5.22E−03 (1.98E−03)+**	1.39E−02 (1.20E−02)+	1.27E+01 (4.46E−01)−	7.57E+00 (2.54E+00)
	F11	3.71E+03 (4.24E+01)−	**1.53E+03 (4.11E+02)+**	2.72E+03 (1.71E+03)−	3.32E+03 (3.43E+02)−	2.46E+03 (5.95E+02)
	F12	5.15E−01 (7.07E−03)−	2.25E−01 (5.03E−02)+	5.94E−01 (2.09E−01)−	5.22E−01 (1.02E−01)−	4.87E−01 (4.77E−02)
	F13	2.53E−01 (3.73E−02)−	2.03E−01 (5.20E−02)−	1.97E−01 (2.34E−02)−	2.02E−01 (1.41E−02)−	**1.77E−01 (1.26E−02)**
	F14	2.69E−01 (4.24E−03)−	2.47E−01 (7.31E−02)−	**1.82E−01 (4.12E−02)+**	2.32E−01 (5.16E−02)−	2.21E−01 (2.77E−02)
	F15	4.97E+00 (7.07E−01)−	3.07E+00 (3.80E−01)+	3.93E+00 (6.02E−01)−	4.19E+00 (1.99E−01)−	3.47E+00 (6.44E−01)
	F16	1.28E+01 (7.21E−01)−	9.29E+00 (2.31E−03)+	9.46E+00 (7.84E−01)+	1.08E+01 (3.53E−01)−	1.05E+01 (6.46E−01)
Hybrid functions	F17	6.21E+04 (2.91E+04)−	1.14E+03 (4.42E+02)−	8.33E+02 (7.68E+02)−	6.45E+01 (7.84E+01)+	3.20E+02 (1.33E+02)
	F18	2.06E+02 (5.66E+00)−	4.84E+01 (7.74E+00)−	4.54E+01 (1.02E+01)−	**1.21E+01 (1.24E+00)+**	2.23E+01 (7.60E+00)
	F19	1.27E+01 (1.13E+00)−	4.05E+00 (6.49E−01)+	5.12E+00 (5.74E−01)−	7.41E+00 (9.43E−02)−	4.35E+00 (6.10E−01)
	F20	5.76E+01 (7.78E+00)−	9.54E+00 (2.37E−01)−	1.19E+01 (4.01E+00)−	1.05E+01 (8.73E−01)−	**7.89E+00 (5.21E+00)**
	F21	5.44E+03 (1.14E+03)−	2.45E+02 (9.25E+01)−	4.08E+02 (1.24E+02)−	4.43E+02 (5.03E+01)−	**1.38E+02 (9.17E+01)**
	F22	2.47E+02 (5.66E+00)−	1.37E+02 (8.93E+00)−	1.69E+02 (7.49E+01)−	1.29E+02 (5.37E+01)−	**1.18E+02 (1.02E+02)**
Composition functions	F23	**3.14E+02 (8.98E−02)**≈	**3.15E+02 (7.53E−14)**≈	**3.14E+02 (1.17E−14)**≈	**3.15E+02 (3.39E−15)**≈	**3.14E+02 (1.72E−15)**
	F24	2.30E+02 (2.82E−01)−	2.27E+02 (3.56E−01)−	**2.23E+02 (1.15E+00)**≈	**2.24E+02 (2.17E−01)**≈	**2.24E+02 (5.74E−01)**
	F25	**2.00E+02 (2.98E−01)**≈	2.05E+02 (7.09E−01)−	**2.00E+02 (4.58E−02)**≈	**2.00E+02 (4.79E−01)**≈	**2.01E+02 (1.74E−01)**
	F26	**1.00E+02 (8.70E−02)**≈	1.04E+02 (6.70E−01)−	**1.00E+02 (4.04E−02)**≈	**1.00E+02 (2.08E−02)**≈	**1.00E+02 (2.07E−02)**
	F27	8.86E+02 (1.84E+01)−	3.75E+02 (1.81E+01)−	3.73E+02 (5.80E+01)−	5.54E+02 (2.31E+02)−	**3.67E+02 (5.16E+01)**
	F28	4.00E+02 (2.42E−01)−	7.60E+02 (8.61E+01)−	3.80E+02 (3.10E+00)−	4.52E+02 (4.53E+01)−	**3.72E+02 (1.45E+00)**
	F29	2.15E+02 (7.07E−01)≈	7.17E+02 (3.49E+00)−	2.53E+02 (7.53E−01)−	4.17E+02 (1.16E+00)−	**2.14E+02 (1.08E+00)**
	F30	5.24E+02 (1.70E+01)−	1.67E+03 (3.03E+02)−	3.98E+02 (3.41E+01)−	4.64E+02 (1.02E+02)−	**2.82E+02 (2.98E+01)**
-/+/≈		23/2/5	19/9/2	18/4/8	20/3/7	

Table 5 Sub-population contribution on CEC2005 benchmark (30D)

Function		Archival sub-population	Exploratory sub-population	Integrated sub-population
F1	30D	**0.47**	0.23	0.30
	50D	**0.51**	0.14	0.35
F2	30D	**0.51**	0.31	0.18
	50D	**0.62**	0.37	0.01
F3	30D	**0.82**	0.17	0.01
	50D	**0.94**	0.05	0.01
F4	30D	**0.52**	0.42	0.06
	50D	**0.80**	0.19	0.01
F5	30D	**0.67**	0.31	0.02
	50D	**0.90**	0.09	0.01
F6	30D	**0.76**	0.22	0.02
	50D	**0.86**	0.13	0.01
F7	30D	**0.48**	0.26	0.26
	50D	**0.53**	0.24	0.23
F8	30D	**0.36**	0.31	0.33
	50D	0.35	0.28	**0.37**
F9	30D	**0.52**	0.15	0.33
	50D	**0.44**	0.19	0.37
F10	30D	0.41	**0.44**	0.15
	50D	**0.41**	0.38	0.21
F11	30D	**0.76**	0.11	0.13
	50D	**0.89**	0.07	0.04
F12	30D	**0.56**	0.28	0.16
	50D	**0.74**	0.08	0.18
F13	30D	**0.73**	0.02	0.25
	50D	**0.88**	0.02	0.10
F14	30D	**0.65**	0.24	0.11
	50D	**0.79**	0.16	0.05
F15	30D	**0.46**	0.24	0.30
	50D	**0.47**	0.21	0.32
F16	30D	**0.45**	**0.46**	0.09
	50D	**0.50**	0.31	0.19
F17	30D	**0.46**	**0.46**	0.08
	50D	**0.52**	0.37	0.11
F18	30D	0.06	0.01	**0.93**
	50D	0.27	0.01	**0.72**
F19	30D	0.05	0.01	**0.94**
	50D	**0.47**	0.28	0.25
F20	30D	0.05	0.02	**0.93**
	50D	**0.41**	0.18	**0.41**
F21	30D	0.35	**0.39**	0.26
	50D	0.16	0.01	**0.83**
F22	30D	0.04	0.03	**0.93**
	50D	0.14	0.01	**0.85**
F23	30D	0.04	0.38	**0.58**
	50D	0.11	0.01	**0.88**
F24	30D	0.08	0.01	**0.91**
	50D	0.36	0.17	**0.47**
F25	30D	**0.51**	0.47	0.02
	50D	**0.49**	0.33	0.18

mal solution, thus reducing the overall convergence speed of the algorithm and the ability to find the optimal solution. Standard MDE-ctd uses dynamic regrouping based on contribution degree. From experiment, both the convergence rate and the ability to find the global optimal solution of Standard MDE-ctd are better than MDE-ctd(c1).

On the mixed function (F15–F25), the performance of MDE-ctd (c1), MDE-ctd (c2) and standard MDE-ctd are similar. This is because there are many local optimizations on highly complex mixed functions. Only the reallocation of computing resources can not make the MDE-ctd jump out of the local optima, and there must be multiple strategies to solve highly complex optimization problems. Therefore, this paper also proposes a multi-population and multi-strategy integration method for MDE-ctd, which can reduce the impact of using multiple mutation strategies on the convergence speed of the algorithm while maintaining population diversity. Table 6, Figs. 5 and 6 shows that the multi-population and multi-strategy integration method used alone has a good effect on the mixed function (F15–F25). Using the multi-population and multi-strategy integration method, MDE-ctd can have powerful search capabilities in the evolutionary process.

Parameter sensitivity

The influence of parameters *ng* and *NP* on the performance of MDE-ctd were analyzed. The different combinations of *ng* and *NP* values were compared on the CEC2005 benchmark function suite and used the mean value to judge the results. The sensitivity analysis results of parameters *ng* and *NP* are shown in Table 7 and Fig. 7.

In parameter sensitivity analysis, when one parameter is analyzed, other parameters are set as standard values (i.e. $ng = 5$ or $NP = 210$). From the analysis results of the parameters *ng* and NP in Table 7 and Fig. 5, it can be found that MDE-ctd is sensitive to the parameters ng and NP of many benchmark functions. If the parameter *ng* is too large, this change can not reflect the search situation in time. If the parameter *ng* is too small, the frequent grouping will make the

Fig. 4 Contribution degree of sub-population to different test functions

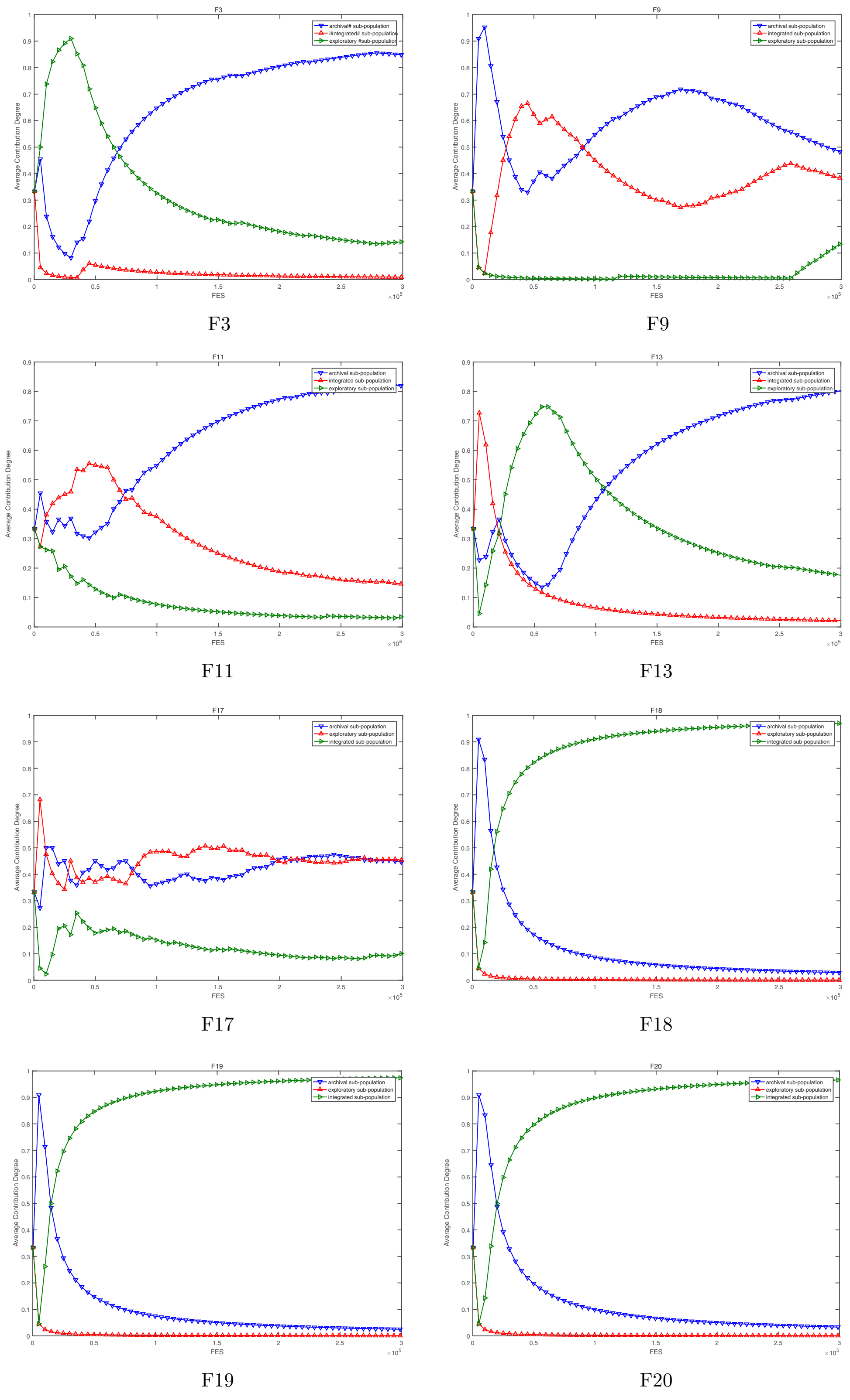


Table 6 Effectiveness of strategys CEC2005 benchmark (30D)

Function	MDE-ctd(c1) mean (std dev)	MDE-ctd(c2) mean (std dev)	MDE-ctd mean (std dev)
F1	**0.00E+00 (0.00E+00)**≈	**0.00E+00 (0.00E+00)**≈	**0.00E+00 (0.00E+00)**
F2	6.02E−19 (9.35E−19)−	1.19E−28 (9.48E−29)−	**7.68E−30 (1.24E−29)**
F3	5.71E+01 (1.68E+01)−	2.72E−01 (2.98E−01)−	**7.12E−02 (5.99E−02)**
F4	6.23E−14 (9.53E−14)−	5.97E−22 (5.76E−22)−	**5.44E−23 (3.95E−23)**
F5	8.48E−06 (6.81E−06)−	2.58E−06 (1.46E−06)−	**6.30E−07 (1.24E−06)**
F6	3.77E+00 (6.54E+00)−	2.20E−06 (2.59E−05)−	**6.02E−16 (4.38E−16)**
F7	3.29E−03 (5.69E−03)−	4.11E−03 (7.11E−03)−	**1.23E−03 (3.89E−03)**
F8	**2.10E+01 (2.08E−03)**≈	**2.10E+01 (1.04E−02)**≈	**2.09E+01 (3.74E−02)**
F9	**0.00E+00 (0.00E+00)**≈	**0.00E+00 (0.00E+00)**≈	**0.00E+00 (0.00E+00)**
F10	4.23E+01 (9.37E+00)−	2.92E+01 (7.20E+00)−	**1.77E+01 (4.97E+00)**
F11	2.74E+01 (1.72E+00)−	**2.57E+01 (5.23E+00)**≈	**2.56E+01 (4.41E−01)**
F12	2.14E+04 (4.46E+03)−	3.17E+03 (6.48E+02)−	**1.29E+03 (7.17E+02)**
F13	2.49E+00 (1.70E−01)−	2.23E+00 (2.60E−01)−	**1.93E+00 (2.55E−01)**
F14	1.29E+01 (1.34E−01)−	1.26E+01 (4.37E−02)−	**1.23E+01 (4.79E−01)**
F15	3.57E+02 (1.07E+02)−	3.73E+02 (3.91E+02)−	**3.14E+02 (9.91E+01)**
F16	6.77E+01 (2.12E−01)−	1.67E+02 (2.03E+02)−	**4.39E+01 (5.89E+00)**
F17	1.24E+02 (1.08E+02)−	4.90E+01 (1.58E+01)−	**4.18E+01 (6.27E+00)**
F18	8.23E+02 (1.15E−02)−	9.04E+02 (1.58E+01)−	**8.16E+02 (2.12E−01)**
F19	8.22E+02 (1.93E−01)−	9.04E+02 (4.05E−01)−	**8.17E+02 (3.96E−01)**
F20	8.22E+02 (8.39E−02)−	9.04E+02 (5.28E−01)−	**8.17E+02 (7.07E−02)**
F21	6.20E+02 (2.08E+02)−	**5.00E+02 (3.36E−09)**≈	**5.00E+02 (4.62E−06)**
F22	**5.00E+02 (1.25E−01)**≈	8.70E+02 (2.59E+01)−	5.02E+02 (7.07E+00)
F23	6.45E+02 (1.93E+02)	**5.34E+02 (7.82E−03)**≈	**5.34E+02 (5.82E−03)**
F24	2.11E+02 (1.91E−01)≈	**2.00E+02 (6.57E−13)+**	2.10E+02 (2.17E−14)
F25	**2.10E+02 (1.87E−01)**≈	**2.09E+02 (2.15E−01)**≈	**2.09E+02 (1.13E−01)**
-/+/≈	19/0/6	17/1/7	

algorithm unable to effectively evaluate the evolution. Compared with the *ng* parameter, the *NP* parameter has a greater impact on the performance of MDE-ctd. Traditionally, it is believed that a larger population is conducive to solving multimodal optimization problems, while a smaller population is conducive to solving single-mode optimization problems. However, the case, which was found in our experiments, is not always unchanged, For example, when dealing with unimodal function F3, the performance of MDE-ctd increases with the increase of population. When dealing with multimodal function F11, the performance of MDE-ctd increases with the decrease in population.

According to the analysis of the experimental data, when $ng = 5$, it can not only reflect the search situation in time, but also reduce the population grouping frequency and effectively evaluate the evolution process. When $\mathrm{NP} = 210$, it can well reduce the impact of population size on the algorithm in the evolution process. Therefore, when $ng = 5$ and $\mathrm{NP} = 210$, the comprehensive performance of MDE-ctd is the best.

Conclusion and future work

This work offered an integrated differential evolution of multiple populations based on contribution degree, known as MDE-ctd, in order to address the deficiencies of the original difference approach in computing resource allocation and handling highly complex optimization issues. In MDE-ctd, three sub-populations (archival sub-population, exploration sub-population, and integrated sub-population) co-evolved at the same time and used a variety of mutation strategies with different advantages to balance exploration and development. In the process of evolution, the contribution of each sub-population to the optimization problem is tracked by the contribution degree, and the dynamic grouping mechanism is adopted to maximize the advantages of different sub-populations and improve the global optimality of MDE-ctd. We effectively realize the dynamic allocation of computing resources among different sub-populations, and allocate more computing resources to the best sub-population in time, thus reducing the waste of computing resources.

Table 7 Computational results of MDE-ctd with different ng and NPsettings over benchmark functions with 30 variables

Function	MDE-ctd ng = 1, NP = 210	MDE-ctd ng = 10, NP = 210	MDE-ctd ng = 20, NP = 210	MDE-ctd ng = 30, NP = 210	MDE-ctd ng = 5, NP = 50	MDE-ctd ng = 5, NP = 100	MDE-ctd ng = 5, NP = 200	MDE-ctd ng = 5, NP = 300	MDE-ctd ng = 5, NP = 210
F1	**0.00E+00**≈	**0.00E+00**≈	**0.00E+00**≈	**0.00E+00**≈	2.10E−29-	**0.00E+00**≈	**0.00E+00**≈	**0.00E+00**≈	**0.00E+00**
F2	4.83E−29−	2.38E−29−	4.50E−29−	1.18E−28−	6.75E−28−	5.12E−28−	5.05E−29−	3.60E−28−	**7.68E−30**
F3	8.33E−02−	**3.16E−03+**	**5.26E−03+**	**2.79E−03+**	4.89E+03−	7.79E+03−	8.23E+02−	**2.53E−07+**	7.12E−02
F4	**1.37E−24+**	9.06E−21−	**6.59E−24+**	2.23E−21−	6.33E−04−	1.99E−10−	8.97E−22−	1.11E−22−	5.44E−23
F5	4.83E−06−	**3.89E−07+**	**1.35E−08+**	6.78E−07−	8.41E+02−	1.19E−02−	**1.08E−07+**	2.05E−06−	6.30E−07
F6	2.19E−12−	5.92E−10−	**3.22E−19+**	5.84E−15−	1.33E+00−	**4.29E−26+**	**3.14E−24+**	1.33E+00−	6.02E−16
F7	3.29E−03−	2.47E−03−	4.92E−03−	2.47E−03−	1.72E−02−	2.62E−03−	6.93E−03−	7.39E−03−	**1.23E−03**
F8	**2.09E+01**≈	**2.09E+01**≈	**2.09E+01**≈	**2.09E+01**≈	**2.10E+01**≈	**2.10E+01**≈	**2.09E+01**≈	**2.09E+01**≈	**2.09E+01**
F9	**0.00E+00**≈	**0.00E+00**≈	**0.00E+00**≈	**0.00E+00**≈	1.33E+00−	**0.00E+00**≈	**0.00E+00**≈	1.70E−01−	**0.00E+00**
F10	2.29E+01−	2.42E+01−	1.82E+01−	2.42E+01−	4.44E+01−	3.58E+01−	2.85E+01−	**1.79E+01**≈	**1.77E+01**
F11	2.63E+01−	**2.49E+01+**	2.60E+01−	2.56E+01≈	**2.36E+01+**	**1.98E+01+**	**2.42E+01+**	2.75E+01−	2.56E+01
F12	3.02E+03−	**2.42E+02+**	3.13E+03−	1.13E+04−	**1.32E+02+**	2.57E+03−	3.24E+03−	1.13E+03−	1.29E+03
F13	1.94E+00≈	2.20E+00−	2.06E+00−	2.18E+00−	**1.88E+00+**	**1.06E+00+**	1.96E+00−	2.24E+00−	1.93E+00
F14	1.24E+01≈	1.24E+01≈	1.25E+01≈	1.23E+01≈	1.26E+01−	**1.20E+01+**	1.21E+01≈	1.29E+01−	1.23E+01
F15	3.33E+02−	4.67E+02−	4.11E+02−	3.33E+02−	3.92E+02−	4.01E+02−	3.33E+02−	3.32E+02−	**3.14E+02**
F16	5.24E+01−	**3.80E+01+**	9.56E+01−	**3.20E+01+**	1.73E+02−	6.23E+01−	7.68E+01−	1.58E+02−	4.39E+01
F17	6.96E+01−	6.58E+01−	4.20E+01≈	5.06E+01−	3.60E+02−	6.07E+01−	1.63E+02−	**4.12E+01+**	4.18E+01
F18	**8.17E+02**≈	**8.17E+02**≈	**8.17E+02**≈	**8.16E+02**≈	**8.19E+02**≈	**8.17E+02**≈	**8.16E+02**≈	**8.16E+02**≈	**8.16E+02**
F19	**8.17E+02**≈	**8.17E+02**≈	**8.16E+02**≈	**8.17E+02**≈	**8.20E+02**≈	8.45E+02−	**8.17E+02**≈	**8.17E+02**≈	**8.17E+02**
F20	**8.17E+02**≈	**8.16E+02**≈	**8.17E+02**≈	**8.16E+02**≈	8.48E+02−	**8.17E+02**≈	**8.17E+02**≈	**8.16E+02**≈	**8.17E+02**
F21	6.21E+02−	6.19E+02−	6.21E+02−	6.20E+02−	**5.00E+02**≈	6.20E+02−	6.19E+02−	6.19E+02−	**5.00E+02**
F22	**5.01E+02**≈	5.05E+02−	**5.04E+02**≈	**5.00E+02**≈	6.41E+02−	7.39E+02−	**5.01E+02**≈	**5.01E+02**≈	**5.02E+02**
F23	6.44E+02−	7.78E+02−	6.45E+02−	**5.34E+02**≈	**5.36E+02**≈	**5.34E+02**≈	6.44E+02−	6.45E+02−	**5.34E+02**
F24	2.11E+02≈	2.11E+02≈	2.11E+02≈	**2.00E+02+**	2.09E+02≈	2.04E+02≈	2.11E+02≈	2.10E+02≈	2.10E+02
F25	**2.10E+02**≈	**2.10E+02**≈	**2.09E+02**≈	**2.09E+02**≈	2.14E+02−	**2.11E+02**≈	**2.10E+02**≈	**2.09E+02**≈	**2.09E+02**
−/+/≈	13/1/11	11/5/9	10/4/11	11/3/11	16/3/6	13/4/8	12/3/10	14/2/9	

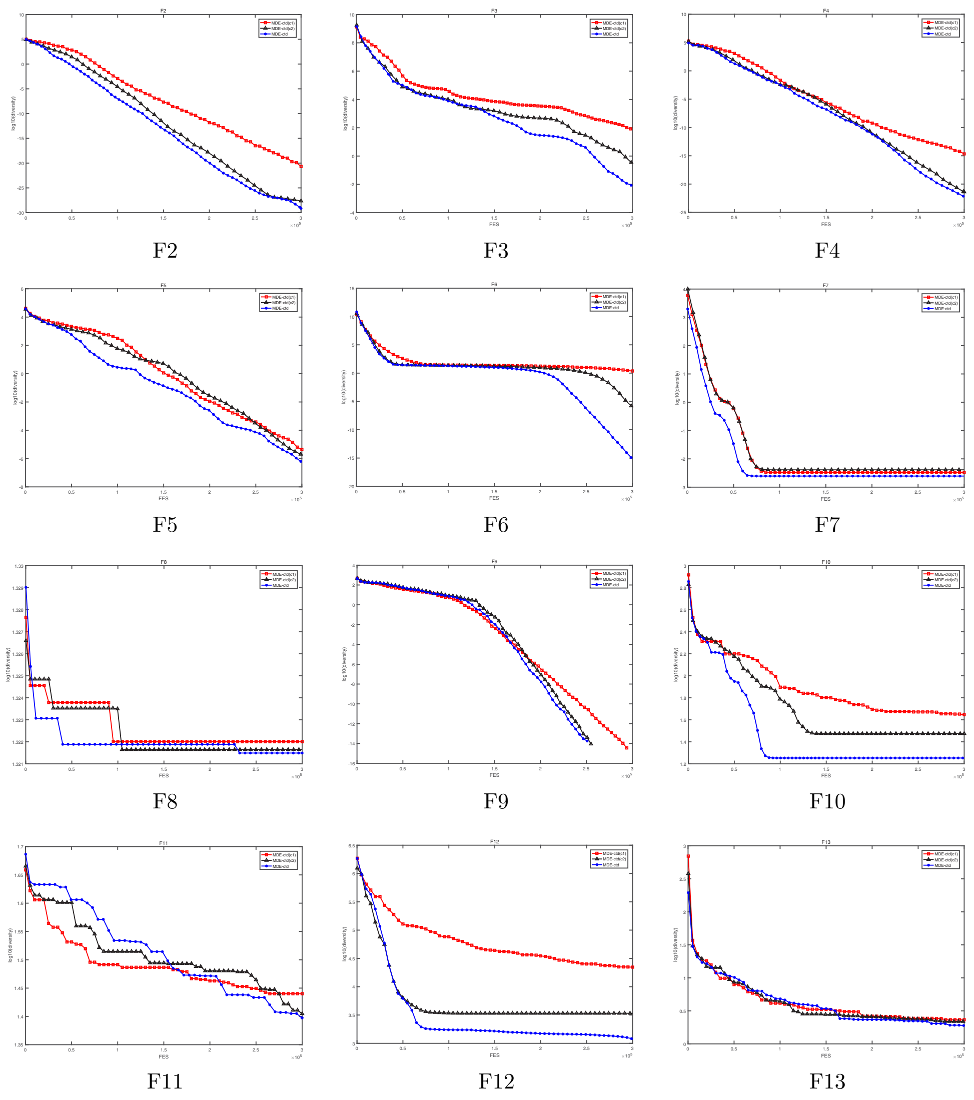
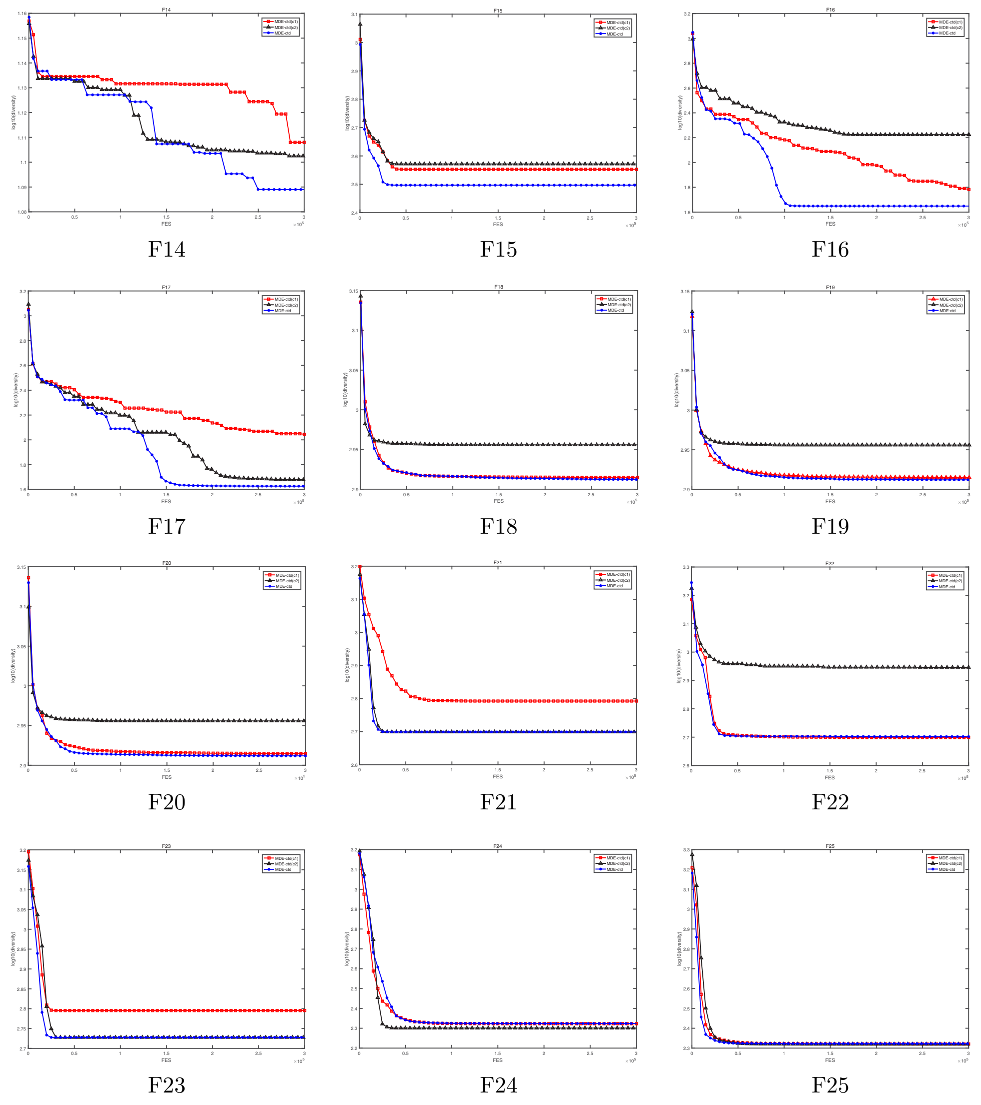


Fig. 5 Convergence curves of MDE-ctd(c1), MDE-ctd(c2), MDE-ctd on CEC2005 benchmark functions F2–F13 with 30D

Fig. 6 Convergence curves of MDE-ctd(c1), MDE-ctd(c2), MDE-ctd on CEC2005 benchmark functions F14–F25 with 30D



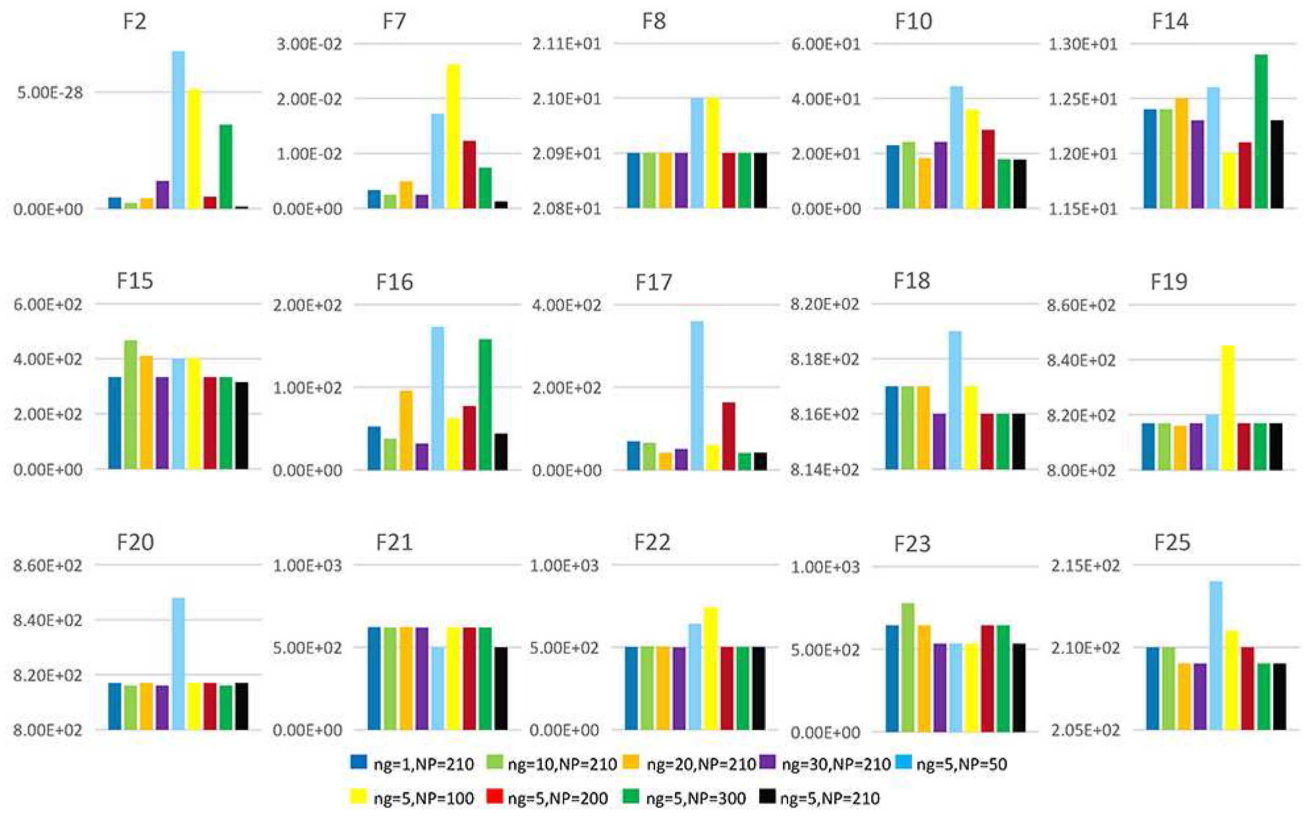


Fig. 7 Bar comparison results of different parameters (ng and NP)

In the following research, the computing resource regrouping method based on contribution degree can provide a new scheme for other multi-population DE algorithms. Meanwhile, MDE-ctd can be applied to some real application problems, such as cloud computing resources scheduling problems, to further test its performance.

Funding Funding was provided by Henan Science and Technology Department (222102320375), Research and Practice Project of Research Teaching Reform in Henan Undergraduate University (2022SYJXLX114), Henan Provincial Science and Technology Research Project (HNKJZK-2023-51B).

Declarations

Conflict of interest All authors certify that they have no affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in this manuscript.

References

1. Storn R (1995) Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical report, International Computer Science Institute, vol 11
2. Mohamed AW (2017) Solving large-scale global optimization problems using enhanced adaptive differential evolution algorithm. Complex Intell Syst 3(4):205–231. https://doi.org/10.1007/s40747-017-0041-0
3. Yang Y, Duan Z (2020) An effective co-evolutionary algorithm based on artificial bee colony and differential evolution for time series predicting optimization. Complex Intell Syst 6(2):299–308. https://doi.org/10.1007/s40747-020-00149-0
4. Jiang LL, Maskell DL, Patra JC (2013) Parameter estimation of solar cells and modules using an improved adaptive differential evolution algorithm. Appl Energy 112:185–193. https://doi.org/10.1016/j.apenergy.2013.06.004
5. Rocca P, Oliveri G, Massa A (2011) Differential evolution as applied to electromagnetics. IEEE Antennas Propag Mag 53(1):38–49. https://doi.org/10.1109/MAP.2011.5773566
6. Wu LH, Wang YN, Yuan XF, Zhou SW (2010) Environmental/economic power dispatch problem using multi-objective differential evolution algorithm. Electric Power Syst Res 80(9):1171–1181. https://doi.org/10.1016/j.epsr.2010.03.010

7. Ji J, Xiao H, Yang C (2021) Hfade-fmd: a hybrid approach of fireworks algorithm and differential evolution strategies for functional module detection in protein-protein interaction networks. Appl Intell 51(2):1118–1132. https://doi.org/10.1007/s10489-020-01791-4
8. Wang Y, Dong W, Dong X (2018) A novel itÖ algorithm for influence maximization in the large-scale social networks. Futur Gener Comput Syst 88:755–763. https://doi.org/10.1016/j.future.2018.04.026
9. Dong WY-F, Dong X-S (2018) Cooperative coevolution with correlation learning between variables for large scale overlapping problem. Acta Electonica Sinica 46(3):529–536
10. Angeline PJ (1995) Adaptive and self-adaptive evolutionary computations. In: Computational intelligence: a dynamic systems perspective. Citeseer
11. Eiben AE, Hinterding R, Michalewicz Z (1999) Parameter control in evolutionary algorithms. IEEE Trans Evol Comput 3(2):124–141. https://doi.org/10.1109/4235.771166
12. Eiben AE, Smith JE et al (2003) Introduction to evolutionary computing, vol 53. Springer, New York
13. Holland JH (1975) Adaptation in natural and artificial systems
14. Brest J, Greiner S, Boskovic B, Mernik M, Zumer V (2006) Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. IEEE Trans Evol Comput 10(6):646–657. https://doi.org/10.1109/TEVC.2006.872133
15. Qin AK, Huang VL, Suganthan PN (2009) Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Trans Evol Comput 13(2):398–417. https://doi.org/10.1109/TEVC.2008.927706
16. Zhang J, Sanderson AC (2009) Jade: adaptive differential evolution with optional external archive. IEEE Trans Evol Comput 13(5):945–958. https://doi.org/10.1109/TEVC.2009.2014613
17. Tanabe R, Fukunaga A (2013) Success-history based parameter adaptation for differential evolution. In: 2013 IEEE congress on evolutionary computation, pp. 71–78. https://doi.org/10.1109/CEC.2013.6557555
18. Wang Z-J, Zhan Z-H, Zhang J (2018) Solving the energy efficient coverage problem in wireless sensor networks: a distributed genetic algorithm approach with hierarchical fitness evaluation. Energies. https://doi.org/10.3390/en11123526
19. Gong Y-J, Chen W-N, Zhan Z-H, Zhang J, Li Y, Zhang Q, Li J-J (2015) Distributed evolutionary algorithms and their models: A survey of the state-of-the-art. Appl Soft Comput 34:286–300. https://doi.org/10.1016/j.asoc.2015.04.061
20. Zhan Z-H, Wang Z-J, Jin H, Zhang J (2020) Adaptive distributed differential evolution. IEEE Trans Cybern 50(11):4633–4647. https://doi.org/10.1109/TCYB.2019.2944873
21. Wu G, Shen X, Li H, Chen H, Lin A, Suganthan PN (2018) Ensemble of differential evolution variants. Inf Sci 423:172–186. https://doi.org/10.1016/j.ins.2017.09.053
22. Li X, Wang L, Jiang Q, Li N (2021) Differential evolution algorithm with multi-population cooperation and multi-strategy integration. Neurocomputing 421:285–302. https://doi.org/10.1016/j.neucom.2020.09.007
23. Zhan Z-H, Liu X-F, Zhang H, Yu Z, Weng J, Li Y, Gu T, Zhang J (2017) Cloudde: a heterogeneous differential evolution algorithm and its distributed cloud version. IEEE Trans Parallel Distrib Syst 28(3):704–716. https://doi.org/10.1109/TPDS.2016.2597826
24. Wu G, Mallipeddi R, Suganthan PN, Wang R, Chen H (2016) Differential evolution with multi-population based ensemble of mutation strategies. Inf Sci 329:329–345. https://doi.org/10.1016/j.ins.2015.09.009. (**Special issue on Discovery Science**)
25. Noman N, Iba H (2008) Accelerating differential evolution using an adaptive local search. IEEE Trans Evol Comput 12(1):107–125. https://doi.org/10.1109/TEVC.2007.895272
26. Peng F, Tang K, Chen G, Yao X (2009) Multi-start jade with knowledge transfer for numerical optimization. In: 2009 IEEE congress on evolutionary computation, pp. 1889–1895. https://doi.org/10.1109/CEC.2009.4983171
27. Li W, Meng X, Huang Y, Fu Z-H (2020) Multipopulation cooperative particle swarm optimization with a mixed mutation strategy. Inf Sci 529:179–196. https://doi.org/10.1016/j.ins.2020.02.034
28. Jia Y-H, Chen W-N, Gu T, Zhang H, Yuan H-Q, Kwong S, Zhang J (2019) Distributed cooperative co-evolution with adaptive computing resource allocation for large scale optimization. IEEE Trans Evol Comput 23(2):188–202. https://doi.org/10.1109/TEVC.2018.2817889
29. Xu P, Luo W, Lin X, Chang Y, Tang K (2022) Difficulty and contribution based cooperative coevolution for large-scale optimization. IEEE Trans Evol Comput. https://doi.org/10.1109/TEVC.2022.3201691
30. Jia Y-H, Mei Y, Zhang M (2022) Contribution-based cooperative co-evolution for nonseparable large-scale problems with overlapping subcomponents. IEEE Trans Cybern 52(6):4246–4259. https://doi.org/10.1109/TCYB.2020.3025577
31. Stanovov V, Akhmedova S, Semenkin E (2019) Selective pressure strategy in differential evolution: exploitation improvement in solving global optimization problems. Swarm Evol Comput 50:100463. https://doi.org/10.1016/j.swevo.2018.10.014
32. Wang Y-J, Zhang J-S (2007) Global optimization by an improved differential evolutionary algorithm. Appl Math Comput 188(1):669–680. https://doi.org/10.1016/j.amc.2006.10.021
33. Zhao S-Z, Suganthan PN (2013) Empirical investigations into the exponential crossover of differential evolutions. Swarm Evol Comput 9:27–36. https://doi.org/10.1016/j.swevo.2012.09.004
34. Wang Y, Cai Z, Zhang Q (2011) Differential evolution with composite trial vector generation strategies and control parameters. IEEE Trans Evol Comput 15(1):55–66. https://doi.org/10.1109/TEVC.2010.2087271
35. Mallipeddi R, Suganthan PN, Pan QK, Tasgetiren MF (2011) Differential evolution algorithm with ensemble of parameters and mutation strategies. Appl Soft Comput 11(2):1679–1696. https://doi.org/10.1016/j.asoc.2010.04.024. (**The Impact of Soft Computing for the Progress of Artificial Intelligence**)
36. Tong L, Dong M, Jing C (2018) An improved multi-population ensemble differential evolution. Neurocomputing 290:130–147. https://doi.org/10.1016/j.neucom.2018.02.038
37. Suganthan PN, Hansen N, Liang JJ, Deb K, Chen Y-P, Auger A, Tiwari S (2005) Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization. KanGAL Rep 2005005(2005):2005
38. Liang JJ, Qu BY, Suganthan PN (2013) Problem definitions and evaluation criteria for the cec 2014 special session and competition on single objective real-parameter numerical optimization. Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore 635:490
39. Fan Q, Wang W, Yan X (2019) Differential evolution algorithm with strategy adaptation and knowledge-based control parameters. Artif Intell Rev 51:219–253