Computers & Operations Research 67 (2016) 155–173

Contents lists available at ScienceDirect

Computers & Operations Research

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

Adaptive differential evolution algorithm with novel mutation strategies in multiple sub-populations

Laizhong Cui, Genghui Li, Qiuzhen Lin*, Jianyong Chen, Nan Lu

College of Computer Science and Software Engineering, Shenzhen University, Shenzhen PR China

ARTICLE INFO

Available online 9 October 2015

Keywords:
Multiple sub-populations
Adaptive parameter control
Replacement strategy
Differential evolution
Global optimization

ABSTRACT

Differential evolution (DE) algorithm has been shown to be a very effective and efficient approach for solving global numerical optimization problems, which attracts a great attention of scientific researchers. Generally, most of DE algorithms only evolve one population by using certain kind of DE operators. However, as observed in nature, the working efficiency can be improved by using the concept of work specialization, in which the entire group should be divided into several sub-groups that are responsible for different tasks according to their capabilities. Inspired by this phenomenon, a novel adaptive multiple sub-populations based DE algorithm is designed in this paper, named MPADE, in which the parent population is split into three sub-populations based on the fitness values and then three novel DE strategies are respectively performed to take on the responsibility for either exploitation or exploration. Furthermore, a simple yet effective adaptive approach is designed for parameter adjustment in the three DE strategies and a replacement strategy is put forward to fully exploit the useful information from the trial vectors and target vectors, which enhance the optimization performance. In order to validate the effectiveness of MPADE, it is tested on 55 benchmark functions and 15 real world problems. When compared with other DE variants, MPADE performs better in most of benchmark problems and real-world problems. Moreover, the impacts of the MPADE components and their parameter sensitivity are also analyzed experimentally.

1. Introduction

With the development of science and technology, there are more and more global optimization problems arising in almost every application field. Generally, a global numerical optimization problem can be defined as follows.

$$\text{Minimize } f(X), X=(x_1,x_2,\ldots,x_D)\in S \tag{1}$$

where $S\subseteq\prod_{i=1}^{D}[x_{i,L},x_{i,U}]$, $-\infty<x_{i,L}<x_{i,U}<+\infty$ for all $i=1,2,\ldots,D$, and the symbol D denotes the dimensions of the decision variables. The target of Eq. (1) is to find a solution $X^*\in S$, which should satisfy the condition that $f(X^*)\le f(X)$ for $\forall X\in S$. Over the last two decades, global optimization problems have attracted a great interest of researchers and numbers of nature-inspired intelligent algorithms have been accordingly proposed for solving global optimization problems, such as genetic algorithm [1], memetic algorithm [2], differential evolution (DE) [3–8], particle swarm optimization [9] and artificial immune algorithm [10]. Among them, DE algorithm is a simple yet effective heuristic algorithm firstly proposed by Storn and Price [3] for dealing with global optimization over continuous space. Due to its outstanding characteristics, such as compact structure, ease to use, speediness and robustness, it has become more and more popular and been extended to handle a variety of optimization problems including multimodal [11], constrained [12], large-scale [13], multi-objective [14], dynamic optimization problems [15] and numbers of real-world applications [16–19].

As a new branch of evolutionary algorithm (EA), DE algorithm shares a similar structure with EA, which includes three important evolutionary operators, *i.e.*, mutation, crossover and selection. The performance of DE primarily relies on these evolutionary operators and their associated parameter settings, such as the scaling factor F and crossover rate Cr. Especially for mutation operator, it generates the mutant vectors to explore the search space and mainly affects the evolutionary direction. Numbers of DE mutation strategies have been proposed and investigated in detail, such as DE/rand/1, DE/rand/2, DE/best/1, DE/best/2, DE/current-to-best/1, DE/current-to-rand/1 [3,20,21] and other variants of them [4–8]. It is experimentally found that different DE mutation strategies have distinct characteristics, which may behave pretty differently in solving various kinds of global optimization problems [22]. For example, when solving unimodal problems, DE/best/1, DE/best/2,

* Corresponding author. Tel.: +86 75526001223; fax: +86 75526534078.
E-mail address: qiuzhlin@szu.edu.cn (Q. Lin).

http://dx.doi.org/10.1016/j.cor.2015.09.006
0305-0548/

DE/rand-to-best/1 and DE/current-to-best/1 are demonstrated to have a fast convergence speed and very good performance, all of which employ the best solution found so far to do further exploration. However, they may lead to premature convergence easily when tackling multimodal problems. On the contrary, DE/rand/1 and DE/rand/2 have a slow convergence speed but strong exploration capability to prevent premature [4]. DE/current-to-rand/1 is a rotation-invariant strategy, which is more suitable for solving the rotated problems than the other DE strategies [21]. On the other hand, different parameter settings also have great impacts on their performances when solving various global optimization problems or even the same problem at different evolutionary stages. For example, a small crossover rate *Cr* in DE is generally suitable for separable functions while a large one is effective for non-separable functions [23]. Generally speaking, a large scaling factor *F* in DE is required at the early stage of the evolution to preserve the population diversity, while a small *F* is preferred to accelerate the population convergence at the later stage. In other words, no any specific parameter setting can always perform very well for all types of global optimization problems [24]. Therefore, in order to effectively use DE for specific optimization problem, a DE strategy is firstly determined and then its corresponding parameter settings are obtained by using a trial-and-error procedure. However, this procedure is very time-consuming and may not be practicable sometime. To overcome this inconvenience, a variety of effective approaches have been proposed to analyze the effects of DE strategies and their associated parameter settings, including the design of new DE strategies [24–29], the adaptive adjustment of multiple DE strategies and their parameter settings [4–7,20], and the combination of multiple DE strategies and parameter settings [8,30,31]. The detailed introductions of these relevant approaches are given in Section 3.

However, when tackling the vast complicated problems in practical applications, in our opinion, the optimization capability of DE is still insufficient as the performance of DE may deteriorate quickly with the increase of dimensionality in search space. This is mainly because it may easily fall into local optimum causing premature convergence or stagnation, especially for solving multimodal problems [32]. Inspired by the working specialization in practical work that different employers with special capabilities are only responsible for certain tasks, this paper proposes a novel adaptive DE algorithm using multiple sub-populations (MPADE) to further enhance the optimization performance. The parent population is divided into three sub-populations based on the individuals' fitness and then each sub-population takes on its search task for either exploitation or exploration. Moreover, an adaptive parameter control strategy is designed for each sub-population to adjust the parameter settings in an online manner and a simple replacement strategy is presented to adequately exploit the useful information from the trial vectors and target vectors. By this way, our proposed algorithm balances very well between exploration and exploitation to enhance the algorithmic robustness and optimization performance. Compared with the existing DE strategies [4–8,31], the contributions of this paper can be summarized as follows.

(1) Based on the individuals' fitness, the parent population is split into three sub-populations as respectively denoted by the inferior, medium, and superior sub-populations. They are evolved using different DE strategies to take on the responsibility of either exploration or exploitation, which is distinctly different from the existing DE strategies. As the inferior one includes the worse individuals that may be far away from the global optimal, DE/current-to-*r*best/2 mutation strategy is correspondingly performed to make a large perturbation on the inferior individual, which undertakes the exploration task. Regarding to the superior sub-population, DE/current-to-*n*best/2 is executed to disturb the better individual by a tiny perturbation, which launches a local search strategy and makes more exploitation of the current area. At last, DE/current-to-*p*best/2 is run on the medium sub-population to achieve a relative balanced strategy between exploration and exploitation.
(2) A simple yet effective adaptive scheme is designed to tune the parameter settings in multiple DE strategies. The corresponding scaling factor *F* and crossover rate *Cr* are dynamically adjusted based on the Cauchy and Gaussian random numbers at each generation. The control parameters of Cauchy and Gaussian distribution are updated based on the successful experience from the former evolutions.
(3) When selecting individuals to survive in the next generation, a replacement strategy is designed, which employs a few best individuals of trial population (offspring population) to replace the same number of the worst individuals in target population (parent population). It is only activated by a small probability that is gradually increased with the generation times. By this way, it can effectively accelerate the convergence speed at the later stage while not losing the population diversity at the beginning.

The advantages of MPADE are also validated by the experimental studies. In order to evaluate the comprehensive performance of MPADE, lots of experiments have been conducted on 55 benchmark problems from CEC2005 [39] and CEC2014 [43] competition on real parameter optimization, which includes various kinds of global optimization problems such as unimodal, multimodal, and complex hybrid composition functions, and 15 real word problems from CEC2011 [40]. When compared with various state-of-the-art DE algorithms (*i.e.*, CoDE [8], EPSDE [31], SaDE [4], jDE [5] and JADE [6], SAMODE [41]), and some recently proposed DE variants (*i.e.*, AEPD-JADE [44], DE-VNS [11], SinDE [42] and rank-jDE [27]), simulation results demonstrate that MPADE performs better on most of benchmark problems and some real world problems. At last, the influences of MPADE components and their parameter sensitivity are also studied experimentally.

The remainder of this paper is organized as follows. Section 2 introduces the basic DE algorithm, while Section 3 gives a review on the relevant research works. The details of our proposed MPADE algorithm are described in Section 4, which includes the sub-population division, the DE mutation strategies, the parameter adaptation scheme, and the replacement strategy. Experimental results of MPADE are presented in Section 5 and compared to various DE variants. Moreover, the effectiveness of MPADE and its parameters sensitivity are analyzed here. At last, Section 6 summarizes the conclusions and future work.

2. Basic differential evolution algorithm

Differential evolution is a branch of EA that follows the general procedures of EA. More specifically, there are three basic operators of DE, including mutation, crossover and selection, as described in the following subsections. In order to describe simply and clearly, some necessary notations and terminologies are introduced at first and then the basic operators of DE algorithm are described.

2.1. Notations and terminologies

Assume that a population with *Np* individuals is denoted by *P* and each individual is represented by the vector $X_i = (x_{1,i}, x_{2,i}, \ldots, x_{D,i})$, where *D* is the number of dimensions in solution space. Since the

population will be varied with the running of evolutionary process, the generation times in DE are expressed by $G=0,1,\ldots,Gmax$, where $Gmax$ is the maximal times of generations. For the ith individual of P at the G generation, it is denoted by $X_i^G=(x_{1,i}^G,x_{2,i}^G,\ldots,x_{D,i}^G)$. The lower and upper bounds in each dimension of search space are respectively recorded by $X_L=(x_{1,L},x_{2,L},\ldots,x_{D,L})$ and $X_U=(x_{1,U},x_{2,U},\ldots,x_{D,U})$. When initialization, the initial population P^0 should be evenly distributed throughout the feasible solution space and the jth component ($j=1,2,\ldots,D$) of ith individuals ($i=1,2,\ldots,Np$) in P^0 is obtained as follows:

$$x_{j,i}^0=x_{j,L}+rand(0,1)\cdot(x_{j,U}-x_{j,L}) \tag{2}$$

where $rand(0,1)$ returns a uniformly distributed random number in [0, 1].

After that, DE turns into a loop of the evolutionary process, including mutation, crossover and selection.

2.2. Mutation strategies

Differential evolution adopts the mutation strategy to generate the mutant vector (donor vector) $V_i^G=(v_1^G,v_2^G,\ldots,v_D^G)$ with respect to each individual X_i^G (called target vector) at generation G. The strategy of DE is generally notated as "DE/x/y/z", where x represents the vector (also called basic vector) to be perturbed, y denotes the number of difference vectors considered for perturbation and z is the crossover scheme (binomial or exponential). According to the published reports [1,14–17], there are six widely used DE mutation strategies, which are described as follows:

$$\mathrm{DE/rand/1}: V_i^G=X_{r1}^G+F\cdot(X_{r2}^G-X_{r3}^G) \tag{3}$$

$$\mathrm{DE/rand/2}: V_i^G=X_{r1}^G+F\cdot(X_{r2}^G-X_{r3}^G)+F\cdot(X_{r4}^G-X_{r5}^G) \tag{4}$$

$$\mathrm{DE/best/1}: V_i^G=X_{best}^G+F\cdot(X_{r1}^G-X_{r2}^G) \tag{5}$$

$$\mathrm{DE/best/2}: V_i^G=X_{best}^G+F\cdot(X_{r1}^G-X_{r2}^G)+F\cdot(X_{r3}^G-X_{r4}^G) \tag{6}$$

$$\mathrm{DE/current-to-rand/1}: V_i^G=X_i^G+F\cdot(X_{r1}^G-X_i^G)+F\cdot(X_{r2}^G-X_{r3}^G) \tag{7}$$

$$\mathrm{DE/current-to-best/1}: V_i^G=X_i^G+F\cdot(X_{best}^G-X_i^G)+F\cdot(X_{r1}^G-X_{r2}^G) \tag{8}$$

In Eqs (3)–(8), $r1,r2,r3,r4,r5$ are the distinct integers randomly generated from [1, NP], which are not equal to i. X_{best}^G is the individual with the best fitness at the generation G. The parameter F is called the scaling factor controlling the mutation scale, which is generally restricted in (0, 1].

2.3. Crossover operator

After mutation, a binomial crossover operator is generally employed on the target vector X_i^G to generate a trial vector (offspring) $U_i^G=(u_{1,i}^G,u_{2,i}^G,\ldots,u_{D,i}^G)$. This is achieved by setting the components of U_i^G as the corresponding components in mutant vector V_i^G or target vector X_i^G, as described by

$$u_{j,i}^G=\begin{cases}v_{j,i}^G & if\ (rand_{j,i}\le Cr \text{ or } j=j_{rand})\\ x_{j,i}^G & otherwise\end{cases} \tag{9}$$

where $rand_{j,i}$ ($i\in[1,Np]$ and $j\in[1,D]$) is a uniformly distributed random number in [0, 1], $Cr\in[0,1]$ is called the crossover rate that controls how many components are inherited from the mutant vector, j_{rand} is a uniformly distributed random integer in [1, D] that makes sure at least one component of trial vector is inherited from the mutant vector.

If the jth component $u_{j,i}^G$ of the trial U_i^G is out of boundary, it will be reset as follows [8]:

$$u_{j,i}^G=\begin{cases}\min\{x_{j,U},2x_{j,L}-u_{j,i}^G\} & if\ u_{j,i}^G<x_{j,L}\\ \max\{x_{j,L},2x_{j,U}-u_{j,i}^G\} & if\ u_{j,i}^G>x_{j,U}\end{cases} \tag{10}$$

2.4. Selection operator

Selection operator determines whether the target or the trial vectors survive and go into the next generation based on their fitness values. Without loss of generality, for a minimization problem, the decision vector with the lower objective value could survive and go into the next generation, which can be defined as follows:

$$X_i^{G+1}=\begin{cases}U_i^G & if\ f(U_i^G)\le f(X_i^G)\\ X_i^G & otherwise\end{cases} \tag{11}$$

where $f(X)$ defines the objective function with the decision variable X and $i=1,2,\ldots,Np$. Obviously, this is a one-to-one greedy selection scheme for finding the global optimum.

3. Relevant works

Over the past 20 years, numbers of research works have been conducted to further improve the performances of DE. Based on the designed DE strategies and their parameter control approaches, the variants of DE algorithms can be generally classified into three categories, such as novel DE strategies, adaptive DE strategies and composite DE strategies. They are briefly introduced in the following subsections respectively.

3.1. Novel DE strategies

A novel trigonometric mutation strategy for DE operation was presented by Fan et al. [25], which is formulated by

$$\begin{aligned}V_i^G=&(X_{r1}^G+X_{r2}^G+X_{r3}^G)/3+(p_2-p_1)\cdot(X_{r1}^G-X_{r2}^G)\\&+(p_3-p_2)\cdot(X_{r2}^G-X_{r3}^G)+(p_1-p_3)\cdot(X_{r3}^G-X_{r1}^G)\end{aligned} \tag{12}$$

where $p_i=|f(X_{ri}^G)|/p'$ and $p'=\sum_{i=1}^3\left|f(X_{ri}^G)\right|$ $(i=1,2,3)$. Obviously, the vector to be perturbed in the trigonometric mutation strategy is not an individual randomly selected from the current population, but is set as the center point of three randomly selected vectors X_{ri}^G $(i=1,2,3)$. The perturbation part is contributed together as the three legs of the triangle and the three weight terms $(p_2-p_1),(p_3-p_2)$ and (p_1-p_3) are adopted to guide the evolutionary direction towards a promising area. This trigonometric mutation operation is integrated with the original DE/rand/1/bin version, which helps to get a better tradeoff between the convergence and the robustness. It is further extended to solve dynamic optimization problems [33], where the experimental results validate its superior performance when compared with other state-of-the-art of DE.

A hybrid DE strategy by utilizing the attraction-repulsion concept of electromagnetism-like algorithm was designed by Kaelo and Ali [28], where the individuals with better fitness values attract others while those with inferior fitness values repel others. The experimental studies reveal that it is significantly better for solving high-dimensional optimization problems. A parameter-free DE algorithm called GBDE was proposed by Wang et al. [29], in which the trial vector is generated by Gaussian distribution to sample the search space based on the target and the best vectors in current generation. It is further enhanced by the hybridization

of GBDE and DE/best/1 [29], which would be helpful to achieve the balance between the exploitation and exploration.

In order to compensate the lack of global exploration ability in DE/current-to-best/1 strategy, a neighborhood-based mutation strategy for DE was presented by Das et al. [34]. This neighborhood-based DE strategy is represented by

$$\begin{cases} L_i^G = X_i^G + \alpha\cdot(X_{nbest,i}^G - X_i^G) + \beta\cdot(X_p^G - X_q^G) \\ g_i^G = X_i^G + \alpha\cdot(X_{gbest,i}^G - X_i^G) + \beta\cdot(X_{r1}^G - X_{r2}^G) \\ V_i^G = w\cdot g_i^G + (1-w)\cdot L_i^G \end{cases} \tag{13}$$

where X_{nbest}^G is the best individual among the neighbors of X_i^G, and X_{gbest}^G is the best vector in the entire population at generation G. X_p^G and X_q^G are two distinct individuals that are randomly selected from the neighbors of X_i^G, while X_{r1}^G and X_{r2}^G are randomly picked from the entire population.

A new DE strategy with optional external archive called DE/current-to-*p*best/1 was introduced by Zhang et al. [6], which exploits both beneficial information from the best solution and other good solutions (usually top $p\%$ individuals, $p \in (0, 100]$). Moreover, the optional external archive digs from the historical data to provide the information of evolutionary direction and maintain the population diversity. A similar DE strategy in MDE-*p*BX, called DE/current-to-gr_best/1, was built by Islam et al. [7], in which a group of the best individuals randomly selected from the current population controls the evolutionary direction. A *p*-best crossover operator is designed by incorporating a greedy parent selection strategy with the conventional binomial crossover. Recently, a variant of the greedy DE/best/1 strategy called DE/lbest/1 strategy was proposed by Yu et al. [24]. In DE/lbest/1, each individual is mutated under the guide of multiple local best individuals instead of one global best individual.

On the other hand, some general DE frameworks were proposed, which can be applied to different DE variants. A novel DE framework based on the proximity characteristics among individuals was designed by Epitropakis et al. [35], where each individual is assigned a selection probability inversely proportional to its distance to the mutant individual. After incorporating this framework with some state-of-the-art DE variants, extensive experiments show that it enhances the optimization performance when solving the majority of the test problems. In [27], a ranking-based DE framework was proposed by Gong and Cai, which is inspired by the natural observation that the superior species always contain good information and they have more chance to guide other species. Some parents for the DE operation are proportionally selected according to their rankings in the current population and thus a higher ranking will get a bigger selection probability. Another DE mutation framework that exploits the information of neighboring individuals was presented by Cai and Wang [26], which aims at exploring the regions of minima and accelerating the convergence rate. The directional information provided by the neighbors is exploited to prevent the individuals from entering an undesired region and guide the mutant individual towards a promising area.

3.2. Adaptive DE strategies

A fuzzy adaptive DE (FADE) was introduced by Liu and Lampinen [36], which adopts fuzzy logic controllers to adjust the parameters F and Cr. In [37], a self-adaptive DE algorithm (SDE) was proposed by Omran et al. in which the crossover rate Cr for each individual is sampled from a normal distribution $N(0.5, 0.15)$. Based on the fact that the better values of control parameters could generate better individuals that are more likely to survive, an efficient adaptation scheme for DE was designed by Brest et al. [5], as follows:

$$F_i^{G+1} = \begin{cases} 0.1 + 0.9\cdot rand_1 & if\ \ rand_2 < \tau_1 \\ F_i^G & otherwise \end{cases} \tag{14}$$

$$Cr_i^{G+1} = \begin{cases} rand_3 & if\ \ rand_4 < \tau_2 \\ Cr_i^G & otherwise \end{cases} \tag{15}$$

where $rand_1$, $rand_2$, $rand_3$ and $rand_4$ are four uniformly distributed random numbers from $[0, 1]$, τ_1 and τ_2 are two predefined parameters that are set to 0.1.

A self-adaptive DE (SaDE) algorithm was proposed by Qin et al. [4], in which the mutation strategies and their associated parameter settings are gradually self-adapted by learning from their previous experiences to generate the promising solutions. The scaling factor F of each target vector is obtained from a normal distribution with mean value 0.5 and standard deviation 0.3, and the crossover rate Cr is generated by $N(Cr_m, 0.1)$ with Cr_m initialized to 0.5. In JADE [6] and MDE-*p*BX [7], the scaling factor and crossover rate are adaptively tuned by Cauchy distribution $Cauchy(F_m, 0.1)$ and Gaussian distribution $Gaussian(F_m,\ 0.1)$ respectively. Success-history based adaptive DE (SHADE) is an improved version of JADE as designed by Tanabe and Fukunaga [45], which proposes a new parameter adaptation mechanism based on the successful searching experience. To further enhance the performance of SHADE, the authors further embedded the Linear Population Size Reduction (LPSR) into SHADE and resultantly a very effective DE variant L-SHADE [46] was proposed. When solving constrained optimization problems, a novel modified DE (NMDE) was presented by Zou et al. [38], in which the scaling factor F and crossover rate Cr for each individual are adapted to a suitable range within the corresponding values of all successful individuals at each generation. More recently, a two-level adaptive parameter control scheme was raised by Yu et al. [24], which is accomplished by two steps. At first, the population-level parameter F_P and Cr_P for the whole population are adaptively determined by the exploration and exploitation states, which are estimated by measuring the population distribution. Secondly, the individual-level parameters F_i and Cr_i for each individual are generated by adjusting F_P and Cr_P according to the individual's fitness value and its distance to the global best individual.

3.3. Composite DE strategies

An ensemble of mutation strategies, crossover strategies and control parameters with DE (EPSDE) was designed by Mallipeddi et al. [30,31,47]. In EPSDE, a pool of distinct DE strategies associated with a pool of values for each control parameter coexists throughout the evolutionary process and competes with each other to produce offspring. Experimental results validate that EPSDE performs better than jDE [5], JADE [6], and SaDE [4] in most of the test problems. A composite DE, called CoDE, was presented by Wang et al. [8], which combines the three well-studied DE strategies (DE/rand/1/bin, DE/rand/2/bin, DE/current-to-rand/1) with three control parameter settings in a random way to generate trial vector. Experimental studies confirm its superior performance over JADE [6], jDE [5], SaDE [4], EPSDE [31]. The above-mentioned composite DE strategies in turn prove that the hybridization of multiple DE strategies have the better performance than single DE strategy. This also encourages us to investigate the performance of multiple DE strategies used in different sub-populations.

4. The proposed algorithm MPADE

In this section, the details of our algorithm MPADE are introduced. In order to clearly describe our algorithm, the algorithmic

flowchart of MPADE is demonstrated in Fig. 1, where *FES* and *max_FES* are respectively the current function evaluation times and maximal function evaluation times, *rand*(0,1) returns a uniformly distributed random number in [0, 1], *G* and *Gmax* respectively indicate the current generations and maximal generations. It is noted that the proposed approaches in this paper are identified with boldface. After initialization, the evolved population is sequentially performed by the procedures of sub-population division, mutation, crossover, selection and replacement strategy. Once the termination condition, usually set to the maximal number of function evaluations (*max_FES*), is satisfied, the best individual is exported as the final result. The proposed approaches, including sub-population division, parameters adaptation method and replacement strategy, are respectively introduced in the following subsections. At last, the pseudo-code of the complete algorithm MPADE is also depicted for further clarification.

4.1. Sub-population division and three novel mutation strategies

Exploitation and exploration are two major tasks of DE. To achieve a better comprehensive performance, most variants of DE have to keep the balance between them. Inspired by the observation that the working efficiency can be greatly improved by using specialization and different employers with special capabilities are responsible only for certain tasks, here in our algorithm, the whole population is divided into three sub-populations based on their fitness values, *i.e.*, the inferior, medium and superior sub-populations. Therefore, the inferior sub-population composed by individuals with worse fitness values, which may apart from the global optimal, undertakes the task of exploration, in which the individuals generally have a large perturbation in order to jump out from the local region and approach towards a more promising region. The superior sub-population including individuals with better fitness values, which may be close to the global optimal, is responsible for the exploitation task. A tiny perturbation is performed in the local area of superior sub-population in order to get a better local or global optimal point. At last, the individuals from the medium sub-population are

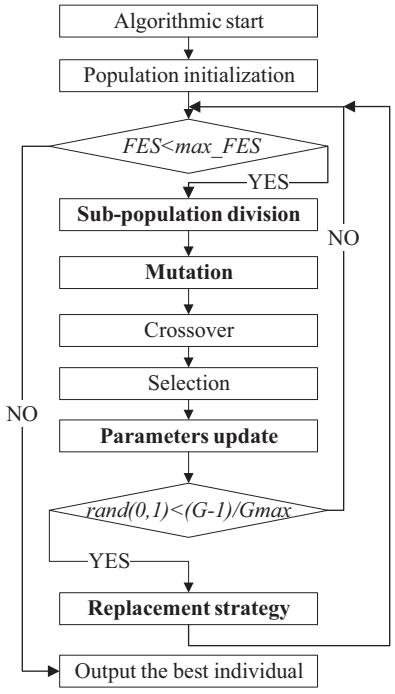

Fig. 1. The algorithmic flowchart of MPADE.

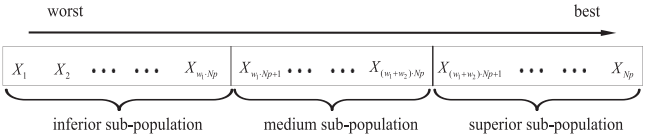

Fig. 2. Illustration of sub-population division.

adopted to balance between exploration and exploitation through the random disturbance during the evolutionary search. This is achieved by adjusting the difference vectors, scaling factor and crossover rate. The detail explanation of our control scheme is given as follows.

First of all, a simple division approach is performed to separate the whole population into three sub-populations based on the fitness values, which are the corresponding objective values as defined in Eq. (1). After all the individuals are sorted in a descending order based on their fitness values, the first $w_1 \cdot Np$ individuals form the inferior sub-population, the subsequent $w_2 \cdot Np$ individuals compose the medium sub-population, and the remainder $w_3 \cdot Np$ individuals belong to the superior sub-population, where $w_1+w_2+w_3=1$ and $w_1, w_2, w_3 \in [0, 1]$. This division approach is simply illustrated in Fig. 2. The selection of w_1, w_2, w_3 will greatly affect the optimization performance, which is analyzed in Section 5.3.

After that, in order to introduce the local neighbor mutation and global remote relative mutation in the corresponding individuals, each individual selects *ns* closest individuals and *rs* farthest individuals based on the Euclidean distance, as its neighbors and remote relatives, respectively. The number of *ns* is linearly reduced, while the value of *rs* is linearly increased with the generations, as defined in Eqs. (16) and (17) respectively.

$$ns = \frac{Np}{10} + ceil\left(\frac{2Np}{5} \cdot \left(1 - \frac{G-1}{Gmax}\right)\right) \tag{16}$$

$$rs = \frac{Np}{10} + ceil\left(\frac{2Np}{5} \cdot \left(\frac{G-1}{Gmax}\right)\right) \tag{17}$$

where *G* and *Gmax* are respectively the current generation and the maximum generation, and *ceil*(*x*) is a ceiling function returning the smallest integer that is greater than its parameter *x*. By this way, *ns* and *rs* are respectively linearly decreased or increased with the generation times in $[Np/10,\ Np/2]$.

Regarding to the inferior sub-population, the DE/current-to-*r*best/2 mutation strategy is correspondingly performed, which is defined as follows:

$$V_i^G = X_i^G + F \cdot (X_{rbest}^G - X_i^G) + F \cdot (X_{r1}^G - X_{r2}^G) + F \cdot (X_{r3}^G - X_{r4}^G) \tag{18}$$

where X_{rbest}^G is the best individual among the relatives of the target vector X_i^G, X_{r1}^G, X_{r2}^G, X_{r3}^G and X_{r4}^G are four different individuals randomly selected from the current population, and all of them are distinct with the target vector X_i^G. Through this strategy, the target vector (the inferior individual) can have a large perturbation that helps to be attracted by a superior individual of the current population. This is because it has a large difference vector $X_{rbest}^G - X_i^G$ and other two random difference vectors. This strategy enables the inferior individuals to explore more in the search space and avoid getting trapped into a local valley. Therefore, it is a less greedy and more explorative variant of the DE/current-to-best/1.

Considering the superior sub-population, the DE/current-to-*n*best/2 strategy is conducted, as defined by

$$V_i^G = X_i^G + F \cdot (X_{nbest}^G - X_i^G) + F \cdot (X_{r1}^G - X_{r2}^G) + F \cdot (X_{r3}^G - X_{r4}^G) \tag{19}$$

where X_{nbest}^G is the best individual in the neighbors of the target vector X_i^G, X_{r1}^G, X_{r2}^G, X_{r3}^G and X_{r4}^G are four random different individuals selected from the neighbors of the target vector, and none of them are equal to the target vector X_i^G. By this strategy, the target

vector X_i^G is attracted by a superior and similar individual of its neighbors, and resultantly a tiny perturbation is added to perform a local search as the selected individuals are very close to each other. Therefore, the superior individuals can sufficiently exploit the current search area to find a local or global optimal solution.

Concerning the medium sub-population, the DE/current-to-*p*best/2 strategy is adopted, which is defined as follows:

$$V_i^G = X_i^G + F \cdot (X_{pbest}^G - X_i^G) + F \cdot (X_{r1}^G - X_{r2}^G) + F \cdot (X_{r3}^G - X_{r4}^G) \quad (20)$$

where X_{pbest}^G is the best individual of the *ps* vectors that are randomly selected from the current population, X_{r1}^G, X_{r2}^G, X_{r3}^G and X_{r4}^G are four different individuals randomly picked from the current population, and all of them are distinct from the target vector X_i^G. DE/current-to-*p*bset/2 is a variant of DE/current-to-*p*best/1, which is a very effective mutation strategy for global numerical optimization, as experimentally validated by JADE [6] and MDE-*p*BX [7]. The parameter *ps* is set to be equal with the parameter *rs*, which is linearly increased with the generation times.

For the parameters setting of *ns* and *rs* in our algorithm, *ns* is linearly decreased while *rs* is linearly increased with the generation times in $[Np/10, Np/2]$. This parameters setting is inspired by Eq. (7) in [7], which controls the proposed algorithm to pay more attention for exploration at the early stage of evolution and exploitation at the late stage of evolution. In the early stage, *ns* is set close to $1/2 \cdot Np$ that ensures that the individuals in superior sub-population have a large number of neighbors and execute a wide range of search, while *rs* is set to $1/10 \cdot Np$, that ensures the individuals in inferior sub-population have a small number of remote relatives and perform a more extensive of search. In the later stage, *ns* is set close to $1/10 \cdot Np$ that makes sure the individuals in superior sub-population have a small number of neighbors and get a small-scale searching, while *rs* is set to $1/2 \cdot Np$, which makes sure the individuals in inferior sub-population has a large number of remote relatives and also executes a small-scale searching relative to the early stage. In addition, the minimal and maximal values of *ns* and *rs* are predefined by analyzing our experimental results and they are insensitive to the optimization performance according to our experiment results.

To make the population quickly converge to the optimal solution, a restriction strategy is set that the first point of the difference vector must be better than the last point. For example, assuming that the difference vector is $X_{r1}^G - X_{r2}^G$, if $f(X_{r1}^G) > f(X_{r2}^G)$ for the minimization problems, the values of X_{r1}^G and X_{r2}^G should be exchanged. Since it is well-known that the difference vectors affect the search direction for a certain extent, the used restriction strategy is expected to offer a higher opportunity that the evolutionary search could move towards a promising direction.

4.2. Parameter adaptation scheme

Generally, most of the traditional DE algorithms use the fixed parameter settings. However, as pointed out in [4–8,31], the performance of DE is very sensitive to the selected DE strategy and the associated control parameters, which also indicates that different parameter settings may be suitable for certain kind of test problems and even be preferred only in certain evolutionary stage for a specific test problem. Therefore, in this paper, a novel parameter adaptation scheme is presented to adjust the scaling factor *F* and the crossover rate *Cr*. Although our parameter adaptation scheme shares some similarities with that used in JADE [6] and MDE-*p*BX [7], our scheme has its own special features that are listed at the end of this subsection.

At each generation *G*, the scaling factor F_i of each individual X_i^G is generated independently based on a Cauchy distribution, which is formulated by

$$F_i = Cauchy(F_m, 0.1) \quad (21)$$

where $Cauchy(F_m, 0.1)$ is a random real number generated according to Cauchy distribution with the location parameter F_m and scaling parameter 0.1. F_i is truncated to be 1 when $F_i > 1$, and F_i is regenerated when $F_i < 0$. All the successful scaling factors at current generation are stored in the set S_F. The location parameter F_m is initialized to be 0.5 and then updated at the end of each generation when S_F is not empty, as defined by

$$F_m^{G+1} = w_F \cdot F_m^G + (1 - w_F) \cdot mean_L(S_F) \quad (22)$$

where $mean_L(\cdot)$ is the Lehmer mean [8] as defined in Eq. (23) and w_F is a random weight factor in [0.8, 1.0] as suggested in [7], which is defined in Eq. (24).

$$mean_L(S_F) = \sum_{F_i \in S_F} F_i^2 / \sum_{F_i \in S_F} F_i \quad (23)$$

$$w_F = 0.8 + 0.2 \cdot rand(0, 1) \quad (24)$$

where *rand*(0,1) is a uniformly generated random number in [0, 1].

However, if the set S_F is empty, it indicates that there is no successful scaling factor in current generation and the parameter F_m^G at this generation *G* is an invalid parameter to a certain extent. In this case, F_m^{G+1} is updated as follows:

$$F_m^{G+1} = C_F \cdot F_m^G + (1 - C_F) \cdot rand(0, 1) \quad (25)$$

$$C_F = 0.5 \cdot rand(0, 1) \quad (26)$$

where *rand*(0,1) is a uniformly generated random number in [0, 1].

For the adaptation in crossover rate, the crossover rate Cr_i of each individual X_i^G is produced independently according to a Gaussian distribution, which is formulated as follows:

$$Cr_i = Gaussian(Cr_m, 0.1) \quad (27)$$

where $Gaussian(Cr_m, 0.1)$ is a random real number generated from a Gaussian distribution with expectation Cr_m and standard deviation 0.1. Cr_i is truncated to be 0 when $Cr_i < 0$ and Cr_i is truncated to be 1 when $Cr_i > 1$. All the successful crossover rates are stored in the set S_{Cr}. The expectation Cr_m is initialized to be 0.5 and then is updated at the end of each generation when S_{Cr} is not empty, as defined by

$$Cr_m^{G+1} = w_{Cr} \cdot Cr_m^G + (1 - w_{Cr}) \cdot mean_L(S_{Cr}) \quad (28)$$

where w_{Cr} is a weight factor randomly generated in [0.8,1.0] and $mean_L(S_{Cr})$ is the Lehmer mean [6] of the set S_{Cr} as similarly calculated by Eq. (23). However, when the set S_{Cr} is empty, Cr_m^{G+1} is updated at the end of each generation, as follows:

$$Cr_m^{G+1} = w_{Cr} \cdot Cr_m^G + (1 - w_{Cr}) \cdot rand(0, 1) \quad (29)$$

$$w_{Cr} = 0.5 \cdot rand(0, 1) \quad (30)$$

Although our adaptive method is inspired by JADE [6] and MDE-*p*BX [7], it has its own specific characteristics, which makes it different from the adaptive approaches in JADE [6] and MDE-*p*BX [7], as clarified below.

1) The Lehmer mean [6] is adopted in our adaptive scheme to update the values of F_m and Cr_m, while JADE employs the usual arithmetic mean to update Cr_m and the Lehmer mean to renew F_m, and MDE-*p*BX [7] only uses the power mean to update the values of F_m and Cr_m.
2) When the sets S_F or S_{Cr} are empty, the parameters F_m and Cr_m would stop updating in JADE and MDE-*p*BX. However, as pointed out in this paper, the empty of sets S_F or S_{Cr} indicates the current parameters F_m and Cr_m are improper at the current

stage and thus they actually need to be changed. Therefore, in our scheme, the values of F_m and Cr_m are varied in a random manner when S_F and S_{Cr} are empty, which is expected to find a more suitable F_m and Cr_m.

3) Compared with only single DE strategy used in JADE and MDE-*p*BX, three DE strategies are coupled with the associated parameters (F_{mw},Cr_{mw} for the inferior sub-population, F_{mm},Cr_{mm} for the medium sub-population and F_{mb},Cr_{mb} for the superior sub-population) in our scheme. Each DE strategy executes the parameter adaptation approach independently.

4.3. Replacement strategy

Although classic DE and many state-of-the-art DE variants adopt one by one selection scheme to keep the population diversity, the beneficial information of the trial vectors and target vectors may not be fully exploited as some excellent individuals do not survive in the next generation. Hence, in order to make good use of trial vectors and target vectors, a direct replacement strategy is presented based on the probability of selection to update the population again.

In order to describe simply and clearly, it is assumed that P^G and U^G respectively denote the target population and the trial population at generation *G*. When the trail vector is better than or equal to the target vector according to their fitness value, i.e., $f(U_i^G) \le f(X_i^G)$, $i = 1, 2, \ldots, Np$, the trail vector and target vector are exchanged instead of replacing. After that, the best *count* trial vectors of trial population U^G are employed to replace the worst *count* target vectors of current target population P^G based on the probability $P_E = (G-1)/Gmax$, which favors exploration at the early stage of the evolution and exploitation during the later stages as P_E is increased gradually . The value of *count* determines the number of individuals that will be replaced, as obtained by

$$count = \lfloor a\% \cdot rand \cdot Np \rfloor \tag{31}$$

where the predefined parameter *a* controls the proportion of population to be replaced. This setting scheme can improve the flexibility of the replacement strategy. In this paper, *a* is fixedly set to 3 according to the extensive experiments. Note that a fixed value can also be assigned to *count*, which should be smaller than $5\% \cdot Np$. Otherwise, excessive exploitation can lead to premature convergence easily according to our experiments.

Our replacement strategy can effectively remove the local optimal solution and accelerate the convergence speed to the global optimal solution at the later stages of evolution. The effectiveness of this replacement strategy will be studied and analyzed in the experimental parts.

Procedures of MPADE

Initialize parameters: Np, max_FES, FES, w_1, w_2, w_3, $G = 1$, $Gmax$
$F_{mw} = F_{mm} = F_{mb} = 0.5$, $Cr_{mw} = Cr_{mm} = Cr_{mb} = 0.5$
Create an initial population P^0 and evaluate P^0
while $FES < max\_FES$
  Sort the evolved population P^G from the worst to the best
  for i=1 to Np
    if $i \le w_1 \cdot Np$ // the inferior sub-population
      $F_i = Cauchy(F_{mw}, 0.1)$, $Cr_i = Gaussian(Cr_{mw}, 0.1)$
      $V_i^G = X_i^G + F_i \cdot (X_{rbest}^G - X_i^G) + F_i \cdot (X_{r1}^G - X_{r2}^G) + F_i \cdot (X_{r3}^G - X_{r4}^G)$
    else if $i \le (w_1 + w_2) \cdot Np$ //the medium sub-population
      $F_i = Cauchy(F_{mm}, 0.1)$, $Cr_i = Gaussian(Cr_{mm}, 0.1)$
      $V_i^G = X_i^G + F_i \cdot (X_{pbest}^G - X_i^G) + F_i \cdot (X_{r1}^G - X_{r2}^G) + F_i \cdot (X_{r3}^G - X_{r4}^G)$
    else //the superior sub-population
      $F_i = Cauchy(F_{mb}, 0.1)$, $Cr_i = Gaussian(Cr_{mb}, 0.1)$
      $V_i^G = X_i^G + F_i \cdot (X_{nbest}^G - X_i^G) + F_i \cdot (X_{r1}^G - X_{r2}^G) + F_i \cdot (X_{r3}^G - X_{r4}^G)$
    end if
    Reset elements of V_i^G if it violates the boundary through Eq. (10)
    Generate trial vector U_i^G through Eq. (9)
    Put trial vector U_i^G into trial population U^G
  end for
  Evaluate the trial population U^G, FES=FES+Np
  for i=1 to Np
    if $f(U_i^G) \le f(X_i^G)$
      Exchange U_i^G and X_i^G, and save the parameters F_i and Cr_i
    end if
  end for
  Update F_{mw}, F_{mm}, F_{mb} and $Cr_{mw}, Cr_{mm}, Cr_{mb}$ using Eqs. (22), (25), (28) and (29)
  if $rand(0,1) \le (G-1)/Gmax$
    $count = \lfloor a\% \cdot rand \cdot Np \rfloor$
    The worst *count* individuals of P^G are replaced by the best *count* individuals of U^G
  end if
  G=G+1
end while
Output the best solution in P^G as the final result

Fig. 3. The complete pseudo-code of MPADE.

4.4. The complete algorithm MPADE

The above subsections have described the three novel algorithmic components of our scheme in detail, *i.e.*, sub-population division and mutation strategies, the parameters adaptation method and the replacement strategy, which are integrated with the framework of classical DE to compose the MPADE algorithm. The pseudo-code of the complete MPADE is demonstrated in Fig. 3. Obviously, after the initialization process, MPADE turns into the loop of sub-population division, mutation, crossover, selection, parameters update and replacement strategy until the number of function evaluations (*FES*) reaches the predefined maximum number of function evaluation (*max_FES*). During the sub-population division phase from lines 05 to 16, all individuals in population P^G are firstly sorted from the worst to the best based on their fitness values. Then, the first $w_1 \cdot Np$ individuals, the medium $w_2 \cdot Np$ individuals and the last $w_3 \cdot Np$ individuals respectively constitute the inferior sub-population, the medium sub-population and the superior sub-population. Different sub-populations employ the corresponding DE strategies coupled with the adaptive parameters F_m and Cr_m to generate the mutant vectors. After mutation, the traditional binary crossover is adopted to generate the trail vectors in line 18. In the selection procedure from lines 22 to 25, a one-to-one greedy selection scheme is adopted to determine whether the target or the trial vectors survive and go into the next generation, and the successful values of F and Cr will be saved in the set S_F and S_{Cr}. Note that different sub-populations have different parameter sets S_F and S_{Cr}. At the end of selection procedure, the parameters F_m (composed by F_{mw}, F_{mm}, F_{mb} for the inferior, medium and superior sub-populations respectively) and Cr_m (composed by Cr_{mw}, Cr_{mm}, Cr_{mb} for the inferior, medium and superior sub-populations respectively) are renewed. At last, if a random event with probability $P_E = (G-1)/Gmax$ occurs, the replacement strategy will be launched. After *FES* reaches *max_FES*, the best individual (solution) in population is reported as the final result.

4.5. The time complexity of MPADE

Compared with the original DE algorithm, the additional computations on the subpopulation division process and replacement process are demanded in MPADE. To evaluate the computational burden caused by the additional operations, the time complexity of MPADE is given. During one generation, first of all, the individuals in the population are sorted according to fitness values, the average complexity of this process is $O(Np \log(Np))$. Then, in order to determine the neighbor and remote relatives, the distance between individuals should be calculated firstly, which takes $Np \cdot (Np-1)/2$ times for calculating the pair wise distance between two different individuals. The complexity of this process is $O(D \cdot Np \cdot (Np-1)/2)$. Secondly, the distances for the individuals in inferior and superior sub-populations are sorted, the corresponding time complexity is $O((ns+rs) \cdot Np \log(Np))$. At last, if the replacement strategy is executed, the average complexity of replacement strategy is $O(2 \cdot Np \log(Np))$ by selecting the *count* best individuals in trial population U^G and the *count* worst individuals in target population. Since the time complexity of the original DE algorithm is $O(Gmax \cdot Np \cdot D)$ [26], the total time complexity of MPADE is $O(Gmax \cdot [Np \log(Np) + D \cdot Np \cdot (Np-1)/2 + (ns+rs) \cdot Np \log(Np) + 2 \cdot Np \log(Np)])$, which is simplified to $O(Gmax \cdot D \cdot Np^2)$.

It should be noted that the recalculation of distance measure in MPADE is performed only when at least one individual of every pair is changed at current generation [26]. Besides that, the distance calculation is only performed for the individuals in the inferior and superior sub-populations. Therefore, the actual complexity of MPADE is far lower than $O(Gmax \cdot D \cdot Np^2)$. Moreover, as pointed out in [26, 35], the additional evaluation cost by computing the pair wise distance between two different individuals and sorting are verified to be negligible when the function evaluation is costly.

5. Experimental results

5.1. Benchmark functions

In this subsection, MPADE is tested by 55 benchmark functions proposed in the special session on real-parameter optimization of CEC 2005 [39] and CEC2014 [43], and 15 real world problems derived from CEC2011 [40]. There are many properties of these test instances, such as unimodality, multimodality, separability, non-separability, rotation and scalability. A detailed description of these benchmark functions and real world problems can be found in [39,40,43].

Table 1
Comparisons of MPADE and five state-of-the-art DE variants on unimodal functions with 30*D*.

Fun	Acceptable accuracy	CoDE	EPSDE	SaDE	jDE	JADE	MPADE
		Mean error (std. dev.)	Mean error (std. dev.)	Mean error (std. dev.)	Mean error (std. dev.)	Mean error (std. dev.)	Mean error (std. dev.)
*F*1	1E−20	1.51E−29 (5.16E−29)	6.73E−30 (3.69E−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)**
SR/Compare/rank		100/≈/1	100/≈/1	**100/≈/1**	**100/≈/1**	**100/≈/1**	**100/\/1**
*F*2	1E-20	4.99E−16 (9.34E−16)	4.65E−25 (1.64E−24)	1.39E−05 (3.35E−05)	1.09E−06 (1.74E−06)	**1.08E−28 (9.37E−29)**	3.94E−28 (1.46E−28)
SR/Compare/rank		0/−/4	100/−/3	0/−/6	0/−/5	**100/≈/1**	100/\/1
*F*3	1E−10	1.39E+05 (1.07E+05)	3.68E+06 (7.86E+06)	4.54E+05 (1.89E+05)	1.65E+05 (8.48E+04)	6.28E+03 (4.87E+03)	**4.89E−17 (3.70E−17)**
SR/Compare/rank		0/−/3	0/−/6	0/−/5	0/−/4	0/−/2	**100/\/1**
*F*4	1E−10	2.02E−01 (7.59E−01)	6.12E+00 (2.00E+01)	2.11E+02 (6.05E+02)	7.80E−02 (2.49E−01)	4.53E−15 (1.39E−14)	**5.67E−28 (2.65E−28)**
SR/Compare/rank		0/−/4	0/−/5	0/−/6	0/−/3	80/−/2	**100/\/1**
*F*5	1E−10	1.42E+03 (5.03E+02)	1.88E+03 (7.97E+02)	3.24E+03 (3.93E+02)	5.10E+02 (4.49E+02)	6.04E−07 (3.29E−06)	**2.23E−11 (1.92E−11)**
SR/Compare/rank		0/−/4	0/−/5	0/−/6	0/−/3	40/−/2	**100/\/1**
−/≈/+		4/1/0	4/1/0	4/1/0	4/1/0	3/2/0	\
Avg-rank		3.2	4	4.8	3.2	1.6	1

"−", "+", and "≈"respectively denote that the performance of the corresponding algorithm is worse than, better than, and similar to that of MPADE according to the Wilcoxon's rank test at a 0.05 significance level.

Table 2
Comparisons of MPADE and five state-of-the-art DE variants on multimodal functions with 30*D*.

Fun	Acceptable accuracy	CoDE	EPSDE	SaDE	jDE	JADE	MPADE
		Mean error (std. dev.)	Mean error (std. dev.)	Mean error (std. dev.)	Mean error (std. dev.)	Mean error (std. dev.)	Mean error (std. dev.)
*F*6	1.00E−10	6.10E+00 (2.10E−10)	3.99E−01 (1.22E+00)	4.48E+01 (3.04E+01)	1.82E+01 (2.05E+01)	1.32E+01 (4.25E+01)	**1.39E−25 (7.54E−25)**
SR/Compare/rank		0/−/3	90/−/2	0/−/6	0/−/5	36.7/−/4	**100/\/1**
*F*7	1.00E−03	1.83E−02 (1.00E−02)	1.81E−02 (1.24E−02)	1.33E−02 (1.23E−02)	1.30E−02 (1.03E−03)	6.73E−03 (7.48E−03)	**1.56E−03 (3.57E−03)**
SR/Compare/rank		6.7/−/5	16.7/−/5	0/−/3	10/−/3	66.7/−/2	**83.3/\/1**
*F*8	2.10E+01	**2.04E+01 (3.08E−01)**	2.10E+01 (4.09E−02)	2.09E+01 (5.05E−02)	2.10E+01 (4.25E−02)	2.09E+01 (4.49E−01)	2.09E+01 (6.57E−02)
SR/Compare/rank		**100/+/1**	83.3/≈/2	86.7/≈/2	86.7/≈/2	86.7/≈/2	80/\/2
*F*9	1.00E−20	**0.00E+00 (0.00E+00)**	6.63E−02 (2.52E−01)	1.66E−01 (3.77E+00)	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**
SR/Compare/rank		**100/≈/1**	93.3/−/5	83.3/−/6	**100/≈/1**	**100/≈/1**	**100/\/1**
*F*10	4.00E+01	4.22E+01 (1.43E+01)	4.92E+01 (1.11E+01)	5.06E+01 (9.52E+01)	5.50E+01 (1.16E+01)	2.68E+01 (8.22E+00)	**2.09E+01 (7.04E+00)**
SR/Compare/rank		46.7/−/3	16.7/−/4	13.3/−/5	16.7/−/6	93.3/−/2	**96.7/\/1**
*F*11	1.50E+01	1.38E+01 (3.02E+00)	3.56E+01 (3.87E+00)	1.63E+01 (2.85E+00)	2.81E+01 (1.61E+00)	2.57E+01 (1.54E+00)	**1.26E+01 (3.61E+00)**
SR/Compare/rank		66.7/−/2	0/−/6	36.7/−/3	0/−/5	0/−/4	**70/\/1**
*F*12	3.00E+03	3.98E+03 (6.09E+03)	3.53E+04 (7.36E+03)	2.97E+03 (2.49E+03)	1.23E+04 (8.55E+03)	5.75E+03 (5.11E+03)	**1.90E+03 (2.24E+03)**
SR/Compare/rank		73.3/−/3	0/−/6	50/−/2	20/−/5	13.3/−/4	**70/\/1**
*F*13	1.50E+00	2.30E+00 (7.87E−01)	1.93E+00 (1.38E−01)	3.87E+00 (3.48E−01)	1.66E+00 (1.65E−01)	1.46E+00 (1.11E−01)	**1.44E+00 (2.27E−01)**
SR/Compare/rank		10/−/5	0/−/4	0/−/6	13.3/−/3	43.3/≈/1	**56.7/\/1**
*F*14	1.30E+01	1.21E+01 (4.52E+00)	1.34E+01 (2.48E−01)	1.27E+01 (2.47E−01)	1.30E+01 (2.43E−01)	1.23E+01 (2.89E−01)	**1.21E+01 (5.60E−01)**
SR/Compare/rank		100/≈/1	3.3/−/6	93.3/−/4	50/−/5	93.3/−/3	**100/\/1**
−/≈/+		6/2/1	8/1/0	8/1/0	7/2/0	6/3/0	\
Avg-rank		2.67	4.44	4.11	3.89	2.56	1.11

"−", "+", and "≈" respectively denote that the performance of the corresponding algorithm is worse than, better than, and similar to that of MPADE according to the Wilcoxon's rank test at a 0.05 significance level.

5.2. Parametric setups of the compared algorithms

In order to evaluate the performance of MPADE, MPADE is compared with some state-of-the-art DE variants, *i.e.*, CoDE [8], EPSDE [31], SaDE [4], jDE [5], and JADE [6] on CEC2005 test functions, CoDE, AEPD-JADE [41], DE_VNS [44], sinDE [42], and rank-jDE [27] on CEC2014 test functions, and CoDE, JADE, SaDE, SAMODE [41] on CEC2011 real world problems. The parameter settings of these compared algorithms in our experiments are the same as recommended in their original papers, which have been well tuned to solve the corresponding benchmark problems. Note that the MATLAB source codes of CoDE, EPSDE, SaDE, jDE, and JADE can be downloaded from the homepage of Qingfu Zhang (http://dces.essex.ac.uk/staff/qzhang/), the source code of AEPDE-JADE is obtained from the original inventor, while the source codes of DE-VNS, sinDE, rank-jDE and SAMODE are all implemented by us. The population size and w_1, w_2, w_3 of MPADE are set to 200 and 0.5, 0.4, 0.1, respectively. The maximal function evaluation (*max_FES*) is adopted as the termination criterion, which is set to 10,000*D* (*D* is the dimension of feasible solution space) for 55 test functions according to the guidelines provided in the special session of CEC 2005 [39] and CEC2014 [43]. For real world problems, *max_FES* is set to 50,000 [40]. For all the compared algorithms, 30 independent runs are conducted on each test function. All the algorithms are implemented in MATLAB2010a and run on a computer with 4-GB RAM Memory and 3.20-GHz CPU. It is noted that the initial population for each test problem are the same in each run of all the algorithms, by setting the same seed for the random number generator. Thus, the differences among their performances could only be ascribed to their internal search capabilities [10].

In our experimental studies, the average and standard deviation of *the function error value* $f(X_{best})-f(X^*)$ are adopted to assess the optimization performance, where X_{best} is the best solution found by the algorithms in each run and X^* is the actual global optimal solution of the test function. Besides that, *Success Rate* (*SR*) is used to represent the percentage of successful runs that can achieve a final function error value below a given threshold value. Wilcoxon's rank-sum test, which is a nonparametric statistic test for the independent samples, is conducted on the experimental results at the 5% significance level to obtain the reliable statistical conclusion. For clarity, the best results for each test problem are marked in boldface.

5.3. Numerical results

5.3.1. Comparison on CEC2005 test functions

In this part, the experiments on all the 25 test functions from CEC2005 are performed with *D* set to 30 and 50. For *D* equal to 100, our experiments will only run on the test functions *F*1–*F*14. When the dimension of search space is 30, the simulation results on unimodal functions *F*1–*F*5, multimodal functions *F*6–*F*14 and hybrid composition function *F*15–*F*25 are respectively presented in Tables 1–3. All the experimental results are obtained from 30 independent runs.

1) *Unimodal functions* *F*1–*F*5: From Table 1, it is clearly found out that MPADE is the best one among the six algorithms on five unimodal functions according to the average rank (Avg-rank) and the *success rate* (*SR*). Considering *F*1, all the compared algorithms solve it very well and obtain the similar results. For *F*2–*F*5, MPADE dramatically outperforms all the compared algorithms

Table 3
Comparisons of MPADE and five state-of-the-art DE variants on hybrid composition functions with 30*D*.

Fun	Acceptable accuracy	CoDE	EPSDE	SaDE	jDE	JADE	MPADE
		Mean error (std. dev.)	Mean error (std. dev.)	Mean error (std. dev.)	Mean error (std. dev.)	Mean error (std. dev.)	Mean error (std. dev.)
*F*15	3.00E+02	3.73E+02 (6.91E+01)	**2.18E+02 (3.72E+01)**	3.71E+02 (6.48E+01)	3.60E+02 (7.70E+01)	3.50E+02 (9.38E+01)	3.77E+02 (6.26E+01)
SR/Compare/rank		10/≈/5	**96.7/+/1**	6.67/+/4	13.3/+/3	13.3/+/2	3.33/\/5
*F*16	1.00E+02	7.65E+01 (6.41E+01)	1.34E+02 (1.01E+01)	8.48E+01 (6.42E+01)	8.07E+01 (2.44E+01)	1.31E+02 (1.49E+02)	**5.05E+01 (1.20E+01)**
SR/Compare/rank		93.3/−/2	56.5/−/6	86.7/−/4	93.3/−/3	50/−/5	**100/\/1**
*F*17	1.00E+02	7.19E+01 (2.66E+01)	1.75E+02 (9.90E+02)	8.70E+01 (6.72E+01)	1.43E+02 (2.51E+01)	1.39E+02 (1.23E+02)	**5.27E+01 (2.43E+01)**
SR/Compare/rank		90/−/2	6.7/−/6	90/−/3	0/−/5	26.7/−/4	**96.7/\/1**
*F*18	9.00E+02	9.06E+02 (1.50E+00)	**8.20E+02 (3.62E+00)**	8.72E+02 (5.99E+01)	9.05E+02 (1.07E+00)	9.04E+02 (6.59E−01)	9.04E+02 (7.67E−01)
SR/Compare/rank		0/−/6	**100/+/1**	40/+/2	0/−/5	0/≈/3	0/\/3
*F*19	9.00E+02	9.06E+02 (1.79E+00)	**8.22E+02 (4.00E+02)**	8.77E+02 (5.95E+01)	9.05E+02 (1.02E+00)	9.04E+02 (8.74E−01)	9.04E+02 (1.29E+00)
SR/Compare/rank		0/−/6	**100/+/1**	36.7/+/2	0/≈/3	0/≈/3	0/\/3
*F*20	9.00E+02	9.07E+02 (2.46E+00)	**8.21E+02 (3.92E+00)**	8.58E+02 (6.29E+01)	9.04E+02 (1.11E+00)	9.04E+02 (9.84E−01)	9.04E+02 (7.80E−01)
SR/Compare/rank		0/−/6	**100/+/1**	53.3/+/2	0/≈/3	0/≈/3	0/\/3
*F*21	5.10E+02	5.30E+02 (9.15E−13)	8.63E+02 (3.28E+00)	5.10E+02 (5.48E+01)	5.00E+02 (2.04E−13)	5.10E+02 (5.48E+01)	**5.00E+02 (1.73E−13)**
SR/Compare/rank		90/−/5	0/−/6	96.7/−/3	100/≈/1	83.3/−/3	**100/\/1**
*F*22	8.50E+02	8.79E+02 (2.34E+01)	**5.16E+02 (5.38E+01)**	9.31E+02 (1.65E+01)	8.75E+02 (1.68E+01)	8.65E+02 (2.01E+01)	8.46E+02 (2.76E+01)
SR/Compare/rank		20/−/5	**100/+/1**	0/−/6	6.67/−/4	46.7/−/3	53.3/\/2
*F*23	5.40E+02	5.48E+02 (7.35E−04)	8.58E+02 (6.13E+01)	5.76E+02 (1.60E+02)	5.34E+02 (3.20E−04)	5.48E+02 (7.35E+01)	**5.34E+02 (5.34E−07)**
SR/Compare/rank		96.7/−/3	3.3/−/6	93.3/−/5	100/≈/1	93.3/−/4	**100/\/1**
*F*24	2.10E+02	**2.00E+02 (2.89E−14)**	2.13E+02 (2.51E+00)	**2.00E+02 (2.89E−14)**	**2.00E+02 (2.89E−14)**	**2.00E+02 (2.89E−14)**	**2.00E+02 (2.89E−14)**
SR/Compare/rank		**100/≈/1**	0/−/6	**100/≈/1**	**100/≈/1**	**100/≈/1**	**100/\/1**
*F*25	2.10E+02	2.12E+02 (1.13E+00)	2.13E+02 (1.81E+00)	2.13E+02 (1.49E+00)	2.10E+02 (5.74E−01)	2.11E+02 (8.35E−01)	**2.09E+00 (3.64E−01)**
SR/Compare/rank		0/−/3	3.3/−/3	0/−/3	23.3/≈/1	76.7/−/3	**96.7/\/1**
−/≈/+		9/2/0	6/0/5	6/1/4	4/6/1	6/4/1	\
Avg-rank		4	3.45	3.18	2.72	3.09	2

"−", "+", and "≈"respectively denote that the performance of the corresponding algorithm is worse than, better than, and similar to that of MPADE according to the Wilcoxon's rank test at a 0.05 significance level.

except that JADE has a similar performance with MPADE on *F*2. Especially for *F*3, only MPADE can obtain a near-global optimal solution. JADE ranks secondly, the promising performance of which mainly benefits from its mutation strategy ("DE/current-to-*p*best/1") and parameter adaptation method. In MPADE, three variants of the mutation strategy DE/current-to-*p*best/1, *i.e.*, DE/current-to-*r*best/2, DE/current-*p*best/2, DE/current-to-*n*best/2, and the parameter adaptation approach similar to JADE are used. Thus, MPADE not only retains the merit of JADE, but also performs even better. On *F*1–*F*5, MPADE performs significantly better than CoDE, EPSDE, SaDE, jDE, and JADE on 4, 4, 4, 4, 3 test functions, respectively. The comprehensive ranking orders on unimodal functions are MPADE, JADE, jDE, CoDE, EPSDE, SaDE in descending direction. It is worth highlighting that the performances of MPADE on *F*3–*F*5 have a remarkable improvement due to its sub-population structure and the associated DE strategies.

2) *Multimodal functions F*6–*F*14: According to Table 2, MPADE is much better than CoDE, EPSDE, SaDE, jDE, and JADE on 6, 8, 8, 7, and 6 test functions respectively. Only CoDE surpasses MPADE on *F*8, whereas MPADE is better than or at least similar with EPSDE, SaDE, jDE and JADE on all the test functions. The comprehensive ranking orders on multimodal functions are MPADE, JADE, CoDE, jDE, SaDE, EPSDE in descending manner. The superior performance of MPADE is achieved mainly because the three sub-populations employ different DE schemes that balance very well between exploration and exploitation. Thus, MPADE can jump out of the local optimal regions, and effectively prevent premature on these 9 multimodal functions.

3) *Hybrid composition functions F*15–*F*25: These test functions are very complicated as each of them is composed by 10 sub-functions with a huge number of local optima [31,47]. To the best of our knowledge, there is no DE variant that is able to find a near-global optimal solution on any test function from *F*15–*F*25. Overall, MPADE is the champion on this group of test functions according to Table 3. It outperforms CoDE, EPSDE, SaDE, jDE, and JADE on 9, 6, 6, 4, and 6 test functions respectively. On the contrary, CoDE, EPSDE, SaDE, jDE and JADE perform better than MPADE on 0, 5, 4, 1, and 1 respectively. It is worth noting that EPSDE performs best on *F*15, *F*18–*F*20 and *F*22, but gives very bad results on the other test functions. The comprehensive ranking orders on this group of hybrid composition functions are MPADE, jDE, JADE, SaDE, EPSDE, CoDE in descending direction. However, MPADE brings no remarkable improvement on these hybrid composition functions.

To summarize, MPADE performs better than the compared algorithms when considering all of the unimodal, multimodal and hybrid composition functions with $D=30$. The convergence curves of the median run (15th) in 30 runs of CoDE, EPSDE, SaDE, jDE,

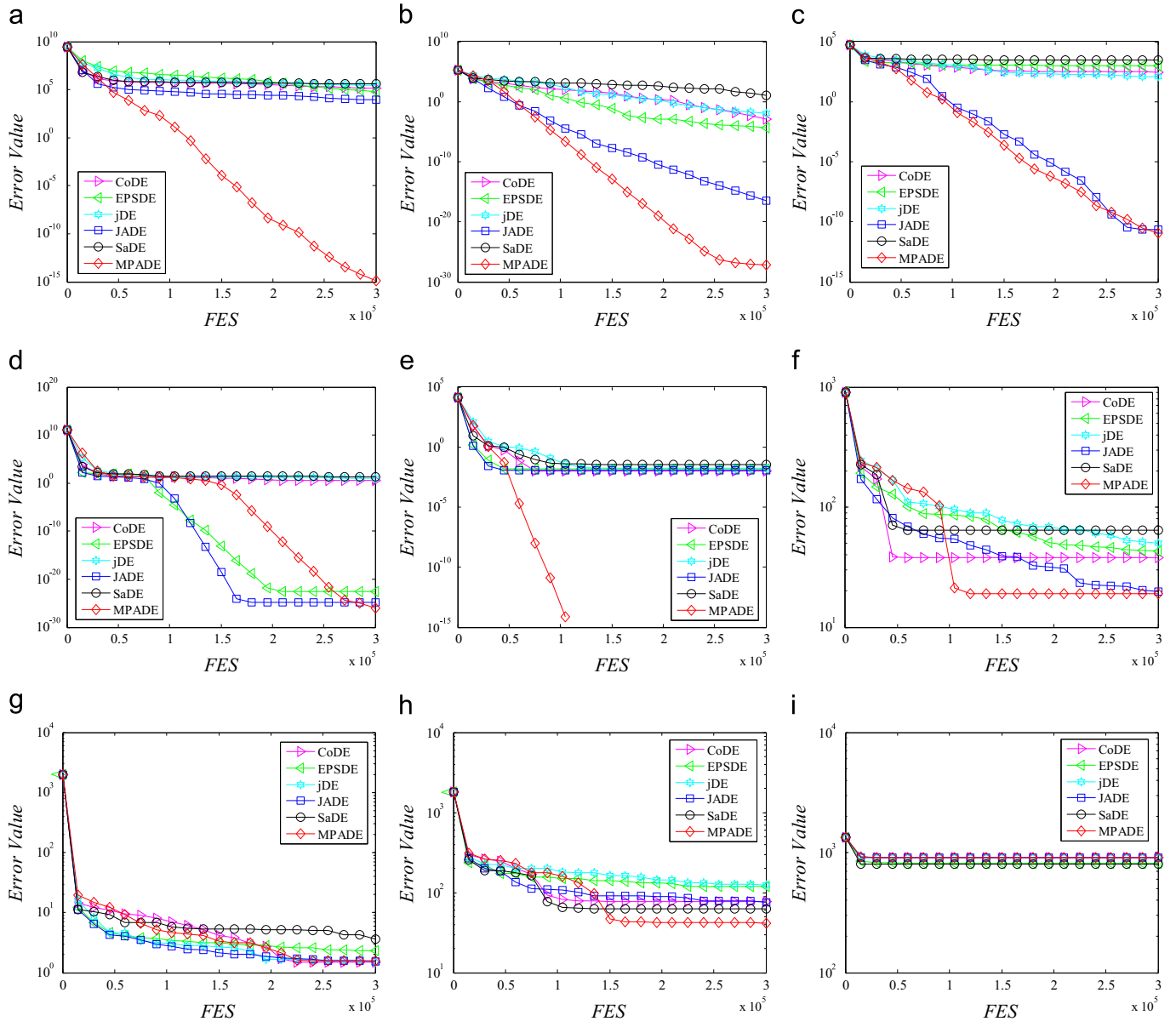

Fig. 4. Convergence curves in the median run of CoDE, EPSDE, SaDE, jDE, JADE, and MPADE on nine test functions with 30*D* (a) *F*3 (b) *F*4 (c) *F*5 (d) *F*6 (e) *F*7 (f) *F*10 (g) *F*13 (h) *F*17 (i) *F*21.

JADE, and MPADE are plotted in Fig. 4 for some selected benchmark functions.

When the dimension of search space D is set to 50, the experimental results are given in Tables 4–6, respectively. All the experimental results are obtained from 30 independent runs.

Considering the unimodal functions *F*1–*F*5 in Table 4, it is pointed out that the performance of MPADE declines on unimodal functions, especially on *F3* when the dimension of the search space is increased to 50. Nonetheless, MPADE still performs better than CoDE, EPSDE, SaDE, jDE and JADE on 5, 4, 4, 4, and 3 functions respectively. In other words, MPADE performs better than or similarly with CoDE, EPSDE, SaDE, jDE on all the test functions. Only JADE surpasses MPADE on *F2*. Although EPSDE has a high success rate on *F2*, it gives a large mean error value, which indicates that the final population of EPSDE may converge prematurely in some runs. In summary, the comprehensive performance of MPADE on unimodal functions is better than that of the other algorithms.

Regarding to the multimodal functions *F*6–*F*14 in Table 5, MPADE performs best on *F*6, *F*9–*F*13, CoDE performs best on *F*8 and *F*14, and JADE performs best on *F7*. When counting all the 9 test functions, MPADE performs better than CoDE, EPSDE, SaDE, jDE, JADE on 7, 8, 8, 6, 5 test functions, respectively. In contrast, CoDE, jDE and JADE perform statistically better than MPADE on 2, 1 and 1 test functions respectively, while EPSDE and SaDE are unable to outperform MPADE on any of 9 test functions. In summary, for *F*6–*F*14 with $D=50$, MPADE is slightly superior to JADE and exceeds the other algorithms significantly.

Concerning the hybrid composition functions *F*15–*F*25 in Table 6, MPADE performs better than CoDE, EPSDE, SaDE, jDE, and JADE on 9, 4 10, 9 and 8 out of 11 test functions, respectively. On the contrary, CoDE, EPSDE, SaDE, jDE, and JADE surpass MPADE on 2, 7, 1, 1, and 2 functions respectively. Based on the average rank in Table 6, MPADE performs best when considering all test functions *F*15–*F*25. It is noted that although EPSDE performs slightly better than MPADE on 7 test functions, it gives worst results on the other

Table 4
Comparisons of MPADE and five state-of-the-art DE variants on unimodal functions with 50*D*.

Fun	Acceptable accuracy	CoDE	EPSDE	SaDE	jDE	JADE	MPADE
		Mean error (std. dev.)	Mean error (std. dev.)	Mean error (std. dev.)	Mean error (std. dev.)	Mean error (std. dev.)	Mean Error (std. dev.)
*F*1	1.00E−20	1.65E−28 (1.92E−28)	1.57E−29 (5.02E−29)	1.51E−29 (5.16E−29)	**0.00E+00 (0.0E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**
SR/Compare/rank		100/−/6	100/≈/1	100/≈/1	**100/≈/1**	**100/≈/1**	**100/\/1**
*F*2	1.00E-20	7.08E−09 (7.14E−09)	3.15E+02 (1.73E+03)	7.24E−02 (6.72E−02)	1.44E−02 (2.10E−02)	**3.68E−27 (3.14E−27)**	1.31E−23 (1.49E−23)
SR/Compare/rank		0/−/3	86.6/−/6	0/−/5	0/−/4	**100/+/1**	100/\/2
*F*3	1.00E+04	3.15E+05 (1.14E+05)	1.32E+07 (3.35E+07)	9.22E+05 (3.25E+05)	4.62E+05 (1.84E+05)	2.77E+04 (8.60E+04)	**2.36E+04 (1.20E+04)**
SR/Compare/rank		0/−/3	0/−/6	0/−/5	0/−/4	6.7/−/2	**10/\/1**
*F*4	1.00E−05	9.32E+02 (9.83E+02)	5.12E+03 (8.61E+03)	6.36E+03 (2.92E+03)	4.03E+02 (1.02E+01)	3.30E−01 (8.12E+01)	**4.26E−06 (1.23E−06)**
SR/Compare/rank		0/−/4	0/−/5	0/−/6	0/−/3	0/−/2	**86.7/\/1**
*F*5	1.00E+00	4.43E+03 (8.04E+02)	4.55E+04 (1.20E+03)	8.36E+03 (1.19E+03)	2.96E+03 (6.44E+02)	1.59E+03 (4.80E+02)	**5.00E+01 (1.13E+02)**
SR/Compare/rank		0/−/4	0/−/6	0/−/5	0/−/3	0/−/2	**10/\/1**
−/≈/+		5/0/0	4/1/0	4/1/0	4/1/0	3/1/1	\
Avg-rank		4	4.8	4.4	3	1.6	1.2

"−", "+", and "≈"respectively denote that the performance of the corresponding algorithm is worse than, better than, and similar to that of MPADE according to the Wilcoxon's rank test at a 0.05 significance level.

Table 5
Comparisons of MPADE and five state-of-the-art DE variants on multimodal functions with 50*D*.

Fun	Acceptable accuracy	CoDE	EPSDE	SaDE	jDE	JADE	MPADE
		Mean error (std. dev.)	Mean error (std. dev.)	Mean error (std. dev.)	Mean error (std. dev.)	Mean error (std. dev.)	Mean error (std. dev)
*F*6	1.00E−10	4.39E+01 (3.42E+01)	6.07E-01 (1.71E+00)	1.01E+02 (9.94E+01)	4.97E+01 (3.12E+01)	1.33E+00 (1.91E+00)	**3.99E−01 (1.22E+00)**
SR/Compare/rank		0/−/4	76.7/−/2	0/−/6	0/−/5	66.7/−/3	**90/\/1**
*F*7	1.00E−10	8.10E−03 (1.31E−02)	7.13E−03 (1.08E−02)	8.19E−03 (1.44E−02)	3.52E−03 (8.96E−03)	**2.14E−03 (4.96E−03)**	4.60E−03 (6.71E−03)
SR/Compare/rank		63.3/−/4	56.7/−/4	63.3/−/4	80/+/2	**80/+/1**	63.3/\/3
*F*8	2.10E+01	**2.05E+01 (3.45E−01)**	2.11E+01 (3.12E−02)	2.11E+01 (3.09E−02)	2.11E+01 (2.56E−02)	2.11E+01 (2.91E−01)	2.11E+01 (6.09E−02)
SR/Compare/rank		**100/+/1**	0/≈/2	0/≈/2	0/≈/2	10/≈/2	6.67/\/2
*F*9	1.00E-10	6.63E−02 (2.52E−01)	1.33E-01 (7.27E−01)	1.89E+00 (1.89E+00)	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**
SR/Compare/rank		93.3/−/4	96.7/−/5	23.3/−/6	**100/≈/1**	**100/≈/1**	**100/\/1**
*F*10	5.00E+01	8.95E+01 (2.09E+01)	1.01E+02 (2.18E+01)	1.27E+02 (2.39E+01)	1.02E+02 (1.50E+01)	4.90E+01 (7.71E+00)	**4.67E+01 (1.05E+01)**
SR/Compare/rank		0/−/3	0/−/4	0/−/6	0/-/5	53.3/−/2	**80/\/1**
*F*11	4.00E+01	3.48E+01 (5.89E+00)	7.01E+01 (4.20E+00)	3.85E+01 (3.83E+00)	5.50E+01 (2.96E+00)	5.14E+01 (2.38E+01)	**3.21E+01 (4.64E+00)**
SR/Compare/rank		76.7/−/2	0/−/6	63.3/−/3	0/−/5	0/−/4	**100/\/1**
*F*12	1.00E+04	2.16E+04 (2.45E+04)	3.11E+05 (4.38E+04)	2.08E+04 (9.09E+03)	1.57E+04 (1.61E+04)	9.87E+04 (1.05E+04)	**1.45E+04 (1.18E+04)**
SR/Compare/rank		40/−/4	0/−/6	50/−/3	46.7/−/2	33.3/−/5	**76.7/\/1**
*F*13	3.00E+00	3.37E+00 (5.92E−01)	6.15E+00 (4.36E−01)	8.63E+00 (1.58E+00)	3.00E+00 (2.24E−01)	2.79E+00 (1.31E−01)	**2.75E+00 (3.10E−01)**
SR/Compare/rank		33.3/−/4	0/−/5	0/−/6	53.3/−/3	90/−/2	**100/\/1**
*F*14	2.20E+01	**2.14E+01 (7.51E−01)**	2.34E+01 (3.37E−01)	2.23E+01 (2.38E−01)	2.27E+01 (2.70E−01)	2.18E+01 (4.01E−01)	2.18E+01 (3.39E-01)
SR/Compare/rank		73.3/+/1	0/−/6	6.67/−/4	0/−/5	76.7/≈/2	63.3/\/2
−/≈/+		7/0/2	8/1/0	8/1/0	6/2/1	5/3/1	\
Avg-rank		3	4.44	4.44	3.33	2.44	1.44

"−", "+", and "≈"respectively denote that the performance of the corresponding algorithm is worse than, better than, and similar to that of MPADE according to the Wilcoxon's rank test at a 0.05 significance level.

test functions. Thus, in some sense, the performance of MPADE is more stable than EPSDE on *F*15–*F*25.

To conclude, the optimization ability of MPADE will deteriorate to a certain extent when the dimension of search space *D* is increased from 30 to 50. However, it still obtains the best comprehensive performance as MPADE is better than CoDE, EPSDE, jDE, SaDE and JADE on 21, 16, 22, 19, 16 out of all 25 test functions respectively, while MPADE is only beaten by CoDE, EPSDE, jDE, SaDE and JADE on 4, 7, 1, 2 and 4 test functions respectively.

When the dimension of solution space *D* is increased to 100, the experimental data are presented in Table 7, where MPADE performs best on 8 out of 14 test functions. MPADE is only beaten

Table 6
Comparisons of MPADE and five state-of-the-art DE variants on hybrid composition functions with 50*D*.

Fun	Acceptable accuracy	CoDE	EPSDE	SaDE	jDE	JADE	MPADE
		Mean error (std. dev.)	Mean error (std. dev.)	Mean error (std. dev.)	Mean error (std. dev.)	Mean error (std. dev.)	Mean error (std. dev.)
*F*15	3.00E+02	3.63E+02 (8.51E+01)	**2.65E+02 (6.87E+01)**	3.73E+02 (6.49E+01)	3.20E+02 (9.61E+02)	3.27E+02 (9.44E+01)	2.83E+02 (9.85E+01)
SR/Compare/rank		20/−/5	70/+/1	13.3/−/6	36.7/−/3	33.3/−/4	56.6/≈/2
*F*16	1.00E+02	9.15E+01 (6.68e+01)	1.43E+02 (6.01E+01)	1.23E+02 (1.13E+02)	8.76E+01 (1.89E+01)	9.86E+01 (1.22E+02)	**7.41E+01 (9.07E+01)**
SR/Compare/rank		83.3/−/3	13.3/−/6	83.3/−/5	96.7/−/2	83.3/−/4	**90/≈/1**
*F*17	1.00E+02	9.34E+01 (6.88E+01)	2.03E+02 (6.21E+01)	1.06E+02 (8.82E+01)	1.81E+02 (2.95E+01)	1.47E+02 (1.09E+02)	**8.91E+01 (1.09E+02)**
SR/Compare/rank		86.7/−/2	0/−/6	76.7/−/3	0/−/5	56.7/−/4	**90/≈/1**
*F*18	9.20E+02	9.25E+02 (2.42E+02)	**8.47E+02 (3.70E+00)**	9.86E+02 (1.55E+01)	9.21E+02 (3.29E+00)	9.23E+02 (6.26E+00)	9.16E+02 (2.94E+00)
SR/Compare/rank		10/−/5	**100/+/1**	0/−/6	40/−/3	26.7/−/4	86.6/≈/2
*F*19	9.20E+02	9.31E+02 (5.88E+00)	**8.46E+02 (3.50E+00)**	9.85E+02 (1.49E+01)	9.21E+02 (3.21E+00)	9.23E+02 (4.23E+00)	9.18E+02 (2.85E+00)
SR/Compare/rank		3.33/−/5	**100/+/1**	0/−/6	43.3/−/3	20/−/4	83.3/≈/2
*F*20	9.20E+02	9.32E+02 (6.51E+00)	**8.45E+02 (3.01E+00)**	9.85E+02 (1.01E+01)	9.20E+02 (3.14E+00)	9.21E+02 (3.39E+00)	9.18E+02 (2.90E+00)
SR/Compare/rank		0/−/5	**100/+/1**	0/−/6	50/−/3	33.3/−/4	76.7/≈/2
*F*21	6.00E+02	5.69E+02 (1.78E+02)	7.30E+02 (3.73E+00)	7.75E+02 (3.44E+02)	7.69E+02 (2.56E+02)	**5.60E+02 (1.60E+02)**	7.52E+02 (2.56E+02)
SR/Compare/rank		86.7/+/2	0/+/3	60/−/6	46.7/−/5	**86.7/+/1**	50/≈/4
*F*22	8.00E+02	9.17E+02 (1.99E+01)	**5.00E+02 (1.92E−01)**	9.82E+02 (1.31E+01)	8.99E+02 (7.33E+00)	9.07E+02 (2.56E+01)	8.72E+02 (1.47E+01)
SR/Compare/rank		0/−/5	**100/+/1**	0/−/6	0/−/3	0/−/4	0/≈/2
*F*23	6.00E+02	6.51E+02 (2.06E+02)	7.33E+02 (3.14E+00)	6.70E+02 (2.66E+02)	7.60E+02 (2.40E+02)	**5.71E+02 (1.20E+02)**	9.17E+02 (1.92E+02)
SR/Compare/rank		76.7/+/2	0/+/4	80/+/3	53.3/+/5	**93.3/+/1**	20/≈/6
*F*24	2.10E+02	2.08E+02 (4.19E+01)	2.36E+02 (1.96E+01)	4.27E+02 (4.18E+02)	**2.00E+02 (1.56E−12)**	**2.00E+02 (1.52E−12)**	**2.00E+00 (1.64E−12)**
SR/Compare/rank		96.7/−/4	10/−/5	76.7/−/6	**100/≈/1**	**100/≈/1**	**100/≈/1**
*F*25	2.15E+02	2.20E+02 (1.89E+00)	2.73E+02 (1.67E+02)	2.22E+02 (2.86E+00)	2.16E+02 (1.55E+00)	2.18E+02 (1.99E+00)	**2.14E+02 (5.70E−01)**
SR/Compare/rank		0/−/4	0/−/6	0/−/5	23.3/−/2	6.67/−/3	**90/≈/1**
−/≈/+		9/0/2	4/0/7	10/0/1	9/1/1	8/1/2	\
Avg-rank		3.82	3.18	5.27	3.18	3.09	2.18

"−", "+", and "≈"respectively denote that the performance of the corresponding algorithm is worse than, better than, and similar to that of MPADE according to the Wilcoxon's rank test at a 0.05 significance level.

by CoDE on *F*8, jDE on *F*7 and *F*9, and JADE on *F*1, *F*2 and *F*6, respectively. Only for *F*8 and *F*9, MPADE gets the third rank in all the compared algorithms. For the other test functions, it receives the first or second ranks according to the optimization performance. It is interesting to point out that jDE can find a global or near global optimal solution on *F*9 and *F*7 in each run, and CoDE performs best on *F*8 in all the cases with $D=30$, 50, 100.

At last, based on the observations from Tables 1–7, it is reasonable to conclude that MPADE is significantly superior to the state-of-the-art DE variants, *i.e.*, CoDE, EPSDE, jDE, SaDE and JADE, on unimodal and multimodal functions, and is also highly competitive on hybrid composition functions.

5.3.2. Comparison on CEC2014 test functions

In this subsection, MPADE is compared with five recently proposed DE variants, *i.e.*, CoDE, AEPD-JADE [44], DE-VNS [11], sinDE [42] and rank-DE [27], on 30 test problems with 30*D* derived from CEC2014 [43]. The experimental results are given in Table 8.

The experimental results in Table 8 clearly demonstrate that MPADE outperforms CoDE, AEPD-JADE, DE-VNS, SinDE and rank-jDE on most of test functions. In more detail, MPADE is respectively better than CoDE, AEPD-JADE, DE-VNS, SinDE and rank-jDE on 16, 16, 19, 15 and 16 test functions. However, CoDE, AEPD-JADE, DE-VNS, SinDE and rank-jDE only outperforms MPADE on 5, 7, 2, 6 and 2 test functions respectively. Obviously, MPADE performs much better than these recently proposed DE variants on unimodal and simply multimodal functions, and competitively on other complex hybrid composition functions.

5.3.3. Comparison on CEC2011 real world problems

This subsection evaluates the performance of MPADE over 15 real-world optimization problems, which are derived from the CEC2011 problems [40]. The performance of MPADE is further compared with CoDE, JADE, SaDE and SAMODE [41]. *max_FES* for these problems is set to 50,000. The parameters in the compared algorithms are set as suggested in their original papers. For all the compared algorithms, 25 runs are conducted independently on each problem, in which the experimental results are collected in Table 9.

Based on the observation in Table 9, MPADE obtains the better results than CoDE, JADE, SaDE and SAMODE on most of real world problems. More specifically, MPADE outperforms CoDE, JADE, SaDE and SAMODE on 7, 8, 8 and 10 problems respectively. On the contrary, CoDE, JADE, SaDE and SAMODE are better than MPADE on 3, 3, 2 and 2 problems respectively. Unfortunately, MPADE is not good at solving the T01FM and T04STR problems. All the compared algorithms obtain the similar performance in tackling the T03BCB, T08TNEP, T11.3ELD, and T11.8HS problems. Except that, MPADE obtains the best performance on the rest of test problems. Therefore, the experimental results of Table 9 clearly demonstrate that MPADE performs better in these real world problems when compared with CoDE, JADE, SaDE and SAMODE.

Table 7
Comparisons of MPADE and five state-of-the-art DE variants on *F*1–*F*14 with 100*D*.

Fun	Acceptable accuracy	CoDE Mean error (std. dev.)	EPSDE Mean error (std. dev.)	SaDE Mean error (std. dev.)	jDE Mean error (std. dev.)	JADE Mean error (std. dev.)	MPADE Mean error (std. dev.)
*F*1	1.00E−20	2.36E−27 (2.14E−27)	5.03E−28 (4.29E−28)	1.12E−28 (1.58E−28)	2.73E−29 (6.17E−29)	**5.47E−30 (1.54E−29)**	6.94E−29 (1.04E−28)
SR/Compare/rank		100/−/6	100/−/4	100/−/4	100/≈/2	**100/+/1**	100/\/2
*F*2	1.00E−10	2.35E−02 (1.60E−02)	1.00E+04 (5.49E+04)	1.37E+02 (6.80E+01)	3.40E+01 (2.43E+01)	**8.25E−15 (1.41E−14)**	1.00E−11 (2.55E−11)
SR/Compare/rank		0/−/3	0/−/6	0/−/5	0/−/4	**100/+/1**	96.7/\/2
*F*3	1.00E+05	2.66E+06 (7.21E+06)	1.46E+06 (3.73E+05)	5.26E+06 (1.01E+06)	2.92E+06 (7.33E+06)	1.89E+05 (8.04E+04)	**1.41E+05 (7.63E+04)**
SR/Compare/rank		0/−/4	0/−/3	0/−/6	0/−/5	6.67/−/2	**30/\/1**
*F*4	1.00E+04	3.87E+04 (9.18E+03)	5.97E+04 (1.80E+04)	6.07E+04 (1.02E+04)	2.76E+04 (5.58E+03)	7.66E+03 (3.45E+03)	**7.01E+03 (7.34E+03)**
SR/Compare/rank		0/−/4	0/−/5	0/−/6	0/−/3	80/−/2	**76.7/\/1**
*F*5	5.00E+03	1.25E+05 (1.60E+03)	1.27E+04 (2.16E+03)	2.66E+04 (2.92E+03)	7.13E+03 (1.09E+03)	3.99E+03 (7.83E+02)	**2.06E+03 (6.88E+02)**
SR/Compare/rank		0/−/6	0/−/4	0/−/5	3.33/−/3	90/−/2	**100/\/1**
*F*6	1.00E−10	1.18E+02 (5.92E+01)	5.11E+01 (4.47E+01)	1.66E+02 (5.90E+01)	8.46E+01 (2.80E+01)	**1.73E+00 (2.01E+00)**	4.26E+00 (5.03E+00)
SR/Compare/rank		0/−/5	0/−/3	0/−/6	0/−/4	**56.6/+/1**	36.7/\/2
*F*7	1.00E−10	6.88E−03 (1.16E−02)	1.17E−02 (2.63E−02)	8.62E−03 (8.63E−03)	**6.14E−16 (2.93E−16)**	4.27E−03 (5.41E−03)	2.88E−03 (4.76E−03)
SR/Compare/rank		50/−/4	60/−/6	36.7/−/5	**100/+/1**	56.6/−/3	70/\/2
*F*8	2.10E+01	**2.01E+01 (8.96E−02)**	2.13E+01 (1.83E−02)	2.13E+01 (2.31E−02)	2.13E+01 (2.47E−02)	2.10E+01 (5.29E−01)	2.12E+01 (2.30E−01)
SR/Compare/rank		**100/+/1**	0/−/4	0/−/4	0/−/4	26.6/+/2	13.3/\/3
*F*9	1.00E−10	7.95E+00 (2.53E+00)	1.01E+02 (3.92E+01)	2.37E+01 (5.42E+00)	**0.00E+00 (0.00E+00)**	1.18E−16 (4.51E−16)	4.31E−01 (7.24E−01)
SR/Compare/rank		0/−/4	0/−/6	0/−/5	**100/+/1**	100/+/2	66.7/\/3
*F*10	1.00E+02	3.50E+02 (6.63E+01)	6.04E+02 (5.28E+01)	5.15E+02 (1.20E+02)	1.54E+02 (1.55E+01)	2.17E+02 (3.61E+01)	**1.35E+02 (3.20E+01)**
SR/Compare/rank		0/−/4	0/−/6	0/−/5	0/−/2	0/−/3	**10/\/1**
*F*11	1.00E+02	1.39E+02 (1.51E+01)	1.61E+02 (1.65E+01)	1.58E+02 (1.76E+00)	1.53E+02 (4.35E+00)	1.30E+02 (1.98E+01)	**9.95E+01 (1.77E+01)**
SR/Compare/rank		3.33/−/3	0/−/6	0/−/5	0/−/4	3.33/−/2	**56.7/\/1**
*F*12	5.00E+04	9.24E+04 (5.33E+04)	5.05E+06 (3.41E+05)	6.23E+04 (3.78E+04)	1.41E+05 (1.13E+05)	6.97E+04 (4.99E+04)	**6.20E+04 (3.99E+04)**
SR/Compare/rank		26.7/−/4	0/−/6	43.3/≈/1	13.3/−/5	23.3/−/3	**36.7/\/1**
*F*13	5.00E+00	8.67E+00 (1.49E+00)	2.98E+01 (1.77E+00)	2.04E+01 (8.54E+00)	6.34E+00 (3.03E−01)	6.66E+00 (4.07E−01)	**6.20E+00 (5.50E−01)**
SR/Compare/rank		0/−/4	0/−/6	0/−/5	0/−/2	3.33/−/3	**3.33/\/1**
*F*14	4.00E+01	4.90E+01 (2.71E−01)	4.95E+01 (3.62E−02)	4.92E+01 (1.17E−01)	4.91E+01 (1.25E−01)	4.89E+01 (1.73E−01)	**4.89E+01 (1.31E−01)**
SR/Compare/rank		0/≈/1	0/−/6	0/−/5	0/−/4	0/≈/1	**0/\/1**
−/≈/+		12/1/1	14/0/0	13/1/0	11/1/2	9/1/4	\
Avg-rank		3.79	5.07	4.79	3.14	2	1.57

"−", "+", and "≈"respectively denote that the performance of the corresponding algorithm is worse than, better than, and similar to that of MPADE according to the Wilcoxon's rank test at a 0.05 significance level.

Table 8
Comparisons of MPADE with five state-of-the-art DE variants on test functions of CEC 2014.

Fuc	Alg					
	CoDE Mean error (std. dev.)	AEPD-JADE Mean error (std. dev.)	DE-VNS Mean error (std. dev.)	sinDE Mean error (std. dev.)	rank-jDE Mean error (std. dev.)	MPADE Mean error (std. dev.)
*F*1	9.64E+03(1.06E+04)−	2.90E+02(5.24E+02)−	4.75E+04(3.52E+04)−	9.05E+05(4.34E+05)−	3.90E+04(3.48E+04)−	**1.65E-14(1.18E−14)**
*F*2	**0.00E+00(0.00E+00)** ≈	5.17E−13(2.36E−12) ≈	**0.00E+00(0.00E+00)** ≈	**0.00E+00(0.00E+00)** ≈	7.59E−15(1.28E−14) ≈	2.08E−14(1.65E−14)
*F*3	**0.00E+00(0.00E+00)** ≈	1.23E−13(1.58E−13) ≈	**0.00E+00(0.00E+00)** ≈	3.61E−11(1.92E−10)−	4.17E−14(2.56E−14) ≈	1.89E−14(2.72E−14)
*F*4	2.69E−02(5.03E−02)−	1.75E−02(6.43E−02)−	1.63E−01(1.21E−04)−	7.48E+00(1.89E+01)−	7.90E−02(2.59E−01)−	**4.16E−14(2.96E−14)**
*F*5	2.06E+01(4.92E−02) ≈	**2.00E+01(1.09E−02)**+	2.06E+01(6.25E−02) ≈	2.05E+01(5.76E−02) ≈	**2.03E+01(2.95E−02)** ≈	2.05E+01(2.25E−01)
*F*6	1.51E+01(8.03E+00)−	4.75E+00(2.40E+00)−	4.31E+00(2.35E+00)−	1.90E+02(1.02E−01)−	2.83E+00(1.89E+00) ≈	**2.78E+00(2.32E+00)**
*F*7	**0.00E+00(0.00E+00)** ≈	2.63E−03(5.88E−03)−	6.31E−03(9.48E−03)−	**0.00E+00(0.00E+00)** ≈	4.93E−04(1.88E−03) ≈	2.65E−14(4.89E−14)
*F*8	1.85E+01(1.88E+00)−	3.79E−15(2.08E−14) ≈	1.21E−03(3.98E−06)−	6.63E−02(2.52E−01)−	**0.00E+00(0.00E+00)** ≈	1.28E−13(1.25E−13)
*F*9	1.27E+02 (1.21E+01)−	3.69E+01(7.59E+00)−	7.93E+01(3.46E+01)−	3.32E+01(7.92E+00)−	4.19E+01(5.03E+00)−	**2.63E+01(6.43E+00)**
*F*10	8.07E+02(8.87E+01)−	**1.12E−01(2.09E−01)**+	2.61E+02(6.19E+01)−	5.08E+00(2.70E+00)−	1.04E−02(1.52E−02)+	4.10E+01(5.59E+01)
*F*11	4.84E+03(1.94E+02)−	**1.43E+03(3.82E+02)**+	4.45E+03(4.06E+02)−	1.88E+03(4.13E+02)+	2.49E+03(2.20E+02)−	2.21E+03(5.30E+02)
*F*12	1.00E+00(1.40E−01)−	**1.30E−01(3.20E−02)**+	8.98E−01(1.11E−01)−	5.30E−01(1.44E−01)−	4.24E−01(4.91E−02)−	**2.02E−01(2.00E−01)**
*F*13	3.88E−01(4.15E−02)−	2.71E−01(5.37E−02)−	3.16E−01(6.53E−02)−	**1.40E−01(3.19E−02)**+	2.58E−01(4.32E−02)−	1.88E−01(2.50E−02)
*F*14	2.76E−01(1.05E+02) ≈	1.57E−01(2.84E−02)+	2.43E−01(4.14E−02)+	**2.11E−01(3.10E−02)**+	2.84E−01(2.24E−02)−	2.73E−01(4.27E−02)
*F*15	1.23E+01 (8.76E−01)−	4.46E+00(1.06E+00)−	9.34E+00(2.08E+00)−	3.50E+00(8.43E−01)−	5.43E+00(4.56E−01)−	**2.99E+00(8.28E−01)**
*F*16	1.16E+01(2.47E−01) ≈	9.74E+00(6.13E−01) ≈	1.11E+01(2.57E−01) ≈	**9.62E+00(5.35E−01)** ≈	9.95E+00(3.25E−01) ≈	9.92E+00(8.11E−01)
*F*17	7.62E+02(2.00E+02)−	1.13E+03(2.66E+02)−	4.12E+03(4.66E+03)−	8.24E+04(6.37E+04)−	1.98E+03(1.86E+03)−	**3.77E+02(1.52E+02)**
*F*18	2.96E+01(6.11E+00)−	2.08E+02(6.21E+02)−	3.50E+01(2.25E+01)−	4.36E+02(8.46E+02)−	5.63E+01(8.20E+01)−	**9.30E+00(3.70E+00)**
*F*19	4.53E+00(2.88E−01)−	4.33E+00(1.05E+00)−	4.06E+00(1.41E+00)−	3.38E+00(7.19E−01)−	4.54E+00(6.68E−01)−	**3.03E+00(6.07E−01)**
*F*20	2.06E+01(3.07E+00)−	6.46E+02(8.98E+02)−	2.33E+01(1.43E+01)−	1.03E+01(3.15E+00)−	2.10E+01(2.08E+01)−	**5.78E+00(1.72E+00)**
*F*21	4.38E+02(1.05E+02)−	2.59E+02(7.35E+01)−	2.80E+02(3.39E+02)−	5.67E+03(4.72E+03)−	3.46E+02(2.08E+02)−	**1.88E+02(1.36E+02)**
*F*22	9.79E+01(4.89E+01)+	1.42E+02(9.32E+01)−	1.28E+02(1.04E+02) ≈	**5.43E+01(5.26E+01)**+	9.01E+01(6.45E+01)+	1.17E+02(8.56E+01)
*F*23	**3.15E+02(1.11E−13)** ≈	**3.15E+02(3.39E−02)** ≈	**3.15E+02(5.78E−14)** ≈	**3.15E+02(5.78E−14)** ≈	**3.15E+02(5.78E−14)** ≈	**3.15E+02(1.73E−13)**
*F*24	**2.14E+02(1.11E+01)**+	2.28E+02(5.60E+00) ≈	2.35E+02(1.02E+01)−	2.22E+02(4.24E+00) ≈	2.27E+02(4.32E+00) ≈	2.24E+02(2.44E+00)
*F*25	**2.00E+02(4.83E−02)**+	2.10E+02(2.72E+00)−	2.08E+02(4.07E+00)−	2.04E+02(4.16E−01) ≈	2.04E+02(9.83E−01) ≈	2.03E+02(1.94E−01)
*F*26	**1.00E+02(4.96E−02)** ≈	**1.00E+02(4.48E−02)** ≈	**1.00E+02(1.82E−02)** ≈	**1.00E+02(2.74E−02)** ≈	**1.00E+02(3.38E−02)** ≈	**1.00E+00(2.74E−02)**
*F*27	3.97E+02(1.83E+01)−	3.82E+02(5.33E+01)−	4.28E+02(5.53E+01)−	**3.00E+02(0.00E+00)** ≈	3.76E+02(5.78E−14)−	3.05E+02(2.05E+01)
*F*28	8.78E+02(4.62E+01) ≈	**6.87E+02(1.47E+02)**+	8.72E+02(1.29E+02) ≈	7.91E+02(2.16E+01)+	8.35E+02(4.31E+01) ≈	8.32E+02(2.96E+01)
*F*29	**4.85E+02(2.88E+02)**+	7.03E+02(3.28E+02)+	6.97E+02(2.04E+02)+	1.31E+03(1.90E+02)−	7.72E+02(9.83E−01)−	7.17E+02(4.30E+00)
*F*30	8.78E+02(1.08E+02)+	1.33E+03(5.03E+02)−	1.06E+03(3.19E+02) ≈	**7.08E+02(1.53E+02)**+	1.86E+03(3.38E−02)−	9.99E+02(4.02E+02)
−/≈/+	**16/9/5**	**16/7/7**	**19/9/2**	**15/9/6**	**16/12/2**	

"−", "+", and "≈" respectively denote that the performance of the corresponding algorithm is worse than, better than, and similar to that of MPADE according to the Wilcoxon's rank test at a 0.05 significance level.

Table 9
Comparisons of MPADE with four state-of-the-art DE variants on real world problems.

Pro	Alg CoDE Mean (std. dev.)	JADE Mean (std. dev.)	SaDE Mean (std. dev.)	SAMODE Mean (std. dev.)	MPADE Mean (std. dev.)
T01FM	**8.26E−01(2.88E+00)**+	1.75E+00(3.58E+00)+	1.01E+00(2.05E+00)+	3.41E+00(2.05E+00) ≈	3.81E+00(5.76E+00)
T02L-J	−1.27E+01(1.84E+00)−	−2.12E+01(1.10E+00)−	−1.48E+01(1.15E+00)−	−1.13E+01(1.92E+00)−	**−2.56E+01(2.56E+00)**
T03BCB	**1.15E−05(0.00E+00)** ≈	**1.15E−05(0.00E+00)** ≈	**1.15E−05(0.00E+00)** ≈	**1.15E−05(0.00E+00)** ≈	**1.15E−05(0.00E+00)**
T04STR	1.43E+01(8.66E+00)+	**1.38E+01(7.68E+00)**+	1.47E+01(8.47E+00)+	1.47E+01(1.08E+00)+	1.54E+01(6.68E+00)
T05Si(B)	−3.04E+01(2.28E+00)−	−3.12E+01(9.65E−01)−	−3.01E+01(1.35E+00)−	−2.41E+01(2.51E+00)−	**−3.26E+01(2.52E+00)**
T06Si(C)	−2.18E+01(1.93E+00)−	**−2.76E+01(1.29E+00)**+	−2.31E+01(1.61E+00) ≈	−2.06E+01(2.91E+00)−	−2.30E+01(2.95E+00)
T07SPRP	1.58E+00(1.22E−01)−	1.25E+00(9.73E−02)−	1.43E+00(1.43E−01)−	1.32E+00(1.91E−01)−	**7.11E−01(1.43E−01)**
T08 TNEP	**2.20E+02(0.00E+00)** ≈	**2.20E+02(0.00E+00)** ≈	**2.20E+02(0.00E+00)** ≈	**2.20E+02(0.00E+00)** ≈	**2.20E+02(0.00E+00)**
T09 LSTP	**1.96E+03(5.64E+02)**+	3.43E+03(7.84E+02)−	2.38E+04(1.14E+04)−	2.29E+05(7.45E+04)−	2.64E+03(8.05E+02)
T10CAAD	−2.13E+01(9.28E−01)−	−2.15E+01(1.70E−01)−	−2.15E+01(1.66E−01)−	−2.05E+01(8.71E−01)−	**−2.16E+01(1.72E−01)**
T11.1DED	5.26E+04(1.06E+00) ≈	5.32E+04(2.89E+03)−	5.84E+04(1.20E+04)−	3.20E+05(1.41E+05)−	**5.25E+04(6.38E+02)**
T11.3ELD	1.54E+04(9.10E+00) ≈	1.54E+04(1.05E+01) ≈	1.54E+04(8.78E−01) ≈	1.54E+04(3.74E−01) ≈	**1.54E+04(2.79E−01)**
T11.8HS	9.43E+05(3.41E+03) ≈	9.44E+05(3.29E+03) ≈	9.43E+05(5.57E+0) ≈	8.95E+05(9.41E+02)+	**9.43E+05(3.25E+03)**
T12(me)	1.88E+01(2.89E+00)−	1.88E+01(1.50E+00)−	2.08E+01(2.14E+00)−	1.83E+01(5.50E+00)−	**1.51E+01(2.10E+00)**
T13(Ca)	1.61E+01(3.62E+00)−	1.95E+01(3.40E+00)−	2.19E+01(2.47E+00)−	1.61E+01(2.81E+00)−	**1.53E+01(4.02E+00)**
−/≈/+	**7/5/3**	**8/4/3**	**8/5/2**	**10/3/2**	

"−", "+", and "≈"respectively denote that the performance of the corresponding algorithm is worse than, better than, and similar to that of MPADE according to the Wilcoxon's rank test at a 0.05 significance level.

5.3.4. Effectiveness of the different components in MPADE

Three algorithmic components are presented in MPADE, *i.e.*, the sub-population division and three DE mutation strategies, the parameters adaptation method and the replacement strategy. In order to demonstrate the effectiveness of each component, some experimental results are provided on *F*2–*F*7, *F*9–*F*14, *F*16, *F*17 from CEC2005 test functions.

(1) *The effectiveness of multiple sub-populations and the associated mutation strategies*: Firstly, to validate the effectiveness of

Table 10
Mean error values of 30 independent runs on *F2*–*F7*, *F9*–*F14*, *F16* and *F17* with 30*D* using CoDE, EPSEE, SaDE, jDE, JADE, ADE_rbest, ADE_pbest, ADE_nbest and MPADE.

Algorithm	Func						
	F2 Mean	*F3* Mean	*F4* Mean	*F5* Mean	*F6* Mean	*F7* Mean	*F9* Mean
CoDE	4.99E−16	1.39E+05	2.02E+01	1.42E+03	6.10E+00	1.83E−02	0.00E+00
EPSDE	4.65E−25	3.68E+06	6.12E+00	1.88E+03	3.99E−01	1.81E−01	6.63E−02
SaDE	1.39E−05	4.54E+05	2.11E+02	3.24E+03	4.48E+01	1.33E−02	1.66E−01
jDE	1.09E−06	1.65E+05	7.80E−02	5.10E+02	1.82E+01	1.30E−02	0.00E+00
JADE	1.08e−28	6.28E+03	4.53E−15	6.04E−07	1.32E+01	6.73E−03	0.00E+00
ADE_nbest	5.51E−11	3.73E+05	6.43E−05	1.14E+03	3.67E+01	5.25E−03	1.96E+01
ADE_pbest	4.70E−28	6.42E−14	7.71E−25	8.22E−10	9.33E−25	3.29E−03	1.68E−11
ADE_rbest	4.70E−28	7.41E−16	7.53E−26	5.07E−10	1.41E−21	4.27E−03	1.57E−12
MPADE	**3.94E−28**	**4.89E−17**	**5.67E−28**	**2.23E−11**	**1.39E−25**	**1.56E−03**	**0.00E+00**

Algorithm	Func						
	F10 Mean	*F11* Mean	*F12* Mean	*F13* Mean	*F14* Mean	*F16* Mean	*F17* Mean
CoDE	4.22E+01	1.38E+01	3.98E+03	2.30E+00	1.21E+01	7.65E+01	7.19E+01
EPSDE	4.92E+01	3.56E−01	3.53E+04	1.93E+00	1.34E+01	1.34E+02	1.75E+02
SaDE	5.06E+01	1.63E+01	2.97E+03	3.87E+00	1.27E+01	8.48E+01	8.70E+01
jDE	5.50E+01	2.81E+01	1.23E+04	1.66E+00	1.30E+01	8.07E+01	1.43E+02
JADE	2.68E+01	2.57E+01	5.75E+03	1.46E+00	1.23E+01	1.31E+02	1.39E+02
ADE_nbest	3.89E+01	2.06E+01	3.44E+03	2.76E+00	1.13E+01	9.30E+01	8.87E+01
ADE_pbest	2.69E+01	2.10E+01	2.42E+03	1.49E+00	1.23E+01	5.19E+01	7.10E+01
ADE_rbest	2.13E+01	1.44E+01	2.28E+03	**1.21E+00**	1.19E+01	5.23E+01	5.68E+01
MPADE	**2.09E+01**	**1.26E+01**	**1.90E+03**	1.44E+00	1.21E+01	**5.05E+01**	**5.27E+01**

multiple sub-populations and the associated mutation strategies in MPADE, the following three variants of MPADE are analyzed.

(1) MPADE without multiple sub-populations and with only one mutation strategy: DE/current-to-*n*best/2, denoted as ADE_*n*best. It is a special case of MPADE with $w_1 = w_2 = 0$ and $w_3 = 1$.
(2) MPADE without multiple sub-populations and with a single mutation strategy: DE/current-to-*p*best/2, denoted as ADE_*p*best. It is a special case of MPADE with $w_1 = w_3 = 0$ and $w_2 = 1$.
(3) MPADE without multiple sub-populations and with an unique mutation strategy: DE/current-to-*r*best/2, denoted as ADE_*r*best. It is a special case of MPADE with $w_2 = w_3 = 0$ and $w_1 = 1$.

The experimental results of MPADE, ADE_*r*best, ADE_*p*best and ADE_*n*best are demonstrated in Table 10, where the best results for each test function are identified with boldface, and the second-best and third-best results are identified with underline. From Table 10, it is found out that ADE_*r*best, and ADE_*p*best also perform better than CoDE, EPSDE, SaDE, jDE, and JADE on most of test functions (*F2*–*F7*, *F10*, *F12*, *F15*, *F16*). This indicates that DE/current-to-*r*best/2 and DE/current-to-*p*best/2 are the two effective mutation operators in MPADE and each of them can perform effectively on most of the test functions. This is mainly because current-to-*r*best/2 and current-to-*p*best/2 can achieve the balance between exploitation and exploration. However, current-to-*n*best/2 is a relatively ineffective mutation operator, when it is employed alone on unimodal functions. This can be attributed to the reason that current-to-*n*best/2 hinders the population from converging to the nearby optimal point. It is obvious that MPADE performs better than ADE_*r*best, ADE_*p*best and ADE_*n*best on most of test functions. These simulations validate the effectiveness of different DE strategies performed on multiple sub-populations as their advantages can be effectively combined to give more promising optimization results when tackling most of test problems.

(2) *The effectiveness of parameter adaptation approach and replacement strategy*: Secondly, our parameter adaptation approach is motivated by JADE [6] and MDE-*p*BX [7], in which the scaling factor F and crossover rate Cr are controlled by the adaptive parameters F_m and Cr_m. The evolutionary trend of F_m and Cr_m on some selected functions in the median run (all the runs are sorted according to the final best error values) are plotted in Fig. 5 (F_{mw}, Cr_{mw} for the inferior sub-population, F_{mm}, Cr_{mm} for the medium sub-population and F_{mb}, Cr_{mb} for the superior sub-population). The curves in Fig. 5 show that $F_m(Cr_m)$ of different sub-populations have similar trend for some test functions, which indicate that different groups with the same F_m and Cr_m may show similar performance. In addition, *F2* needs a large value of Cr_m and a small value of F_m at the early stage of evolution but prefers a large value of F_m in late stage. The parameter Cr_m is close to 0.9 and F_m is limited between 0.2 and 0.5, which are suitable for *F3*, *F4*, *F5* and *F6*. However, for *F10*, a fixed value or a small range of F_m and Cr_m could not be found, as the evolutionary curves seem very random in certain evolutionary stages.

In order to demonstrate the superiority of the proposed parameter adaptation scheme in MPADE, three variants of MPADE are introduced for comparison:

(1) MPADE without the parameter adaptation approach, and set $F_{mw} = F_{mm} = F_{mb} = 0.5$ and $Cr_{mw} = Cr_{mm} = Cr_{mb} = 0.9$ throughout the evolutionary process, denoted as MPDE (non-adaptive MPADE).
(2) MPADE with the parameter adaptation method proposed in JADE [6], denoted as JA-MPADE.
(3) MPADE with the parameter adaptation approach presented in MDE-*p*BX [7], denoted as M-MPADE.

Moreover, to validate the effectiveness of our replacement strategy, the following variant of MPADE is put forward.

(4) MPADE without the replacement strategy, denoted as MPADE-WRS.

The comparisons of MPADE with MPDE, JA-MPADE, M-MPADE, and MPADE-WRS are given in Table 11. Obviously, MPADE

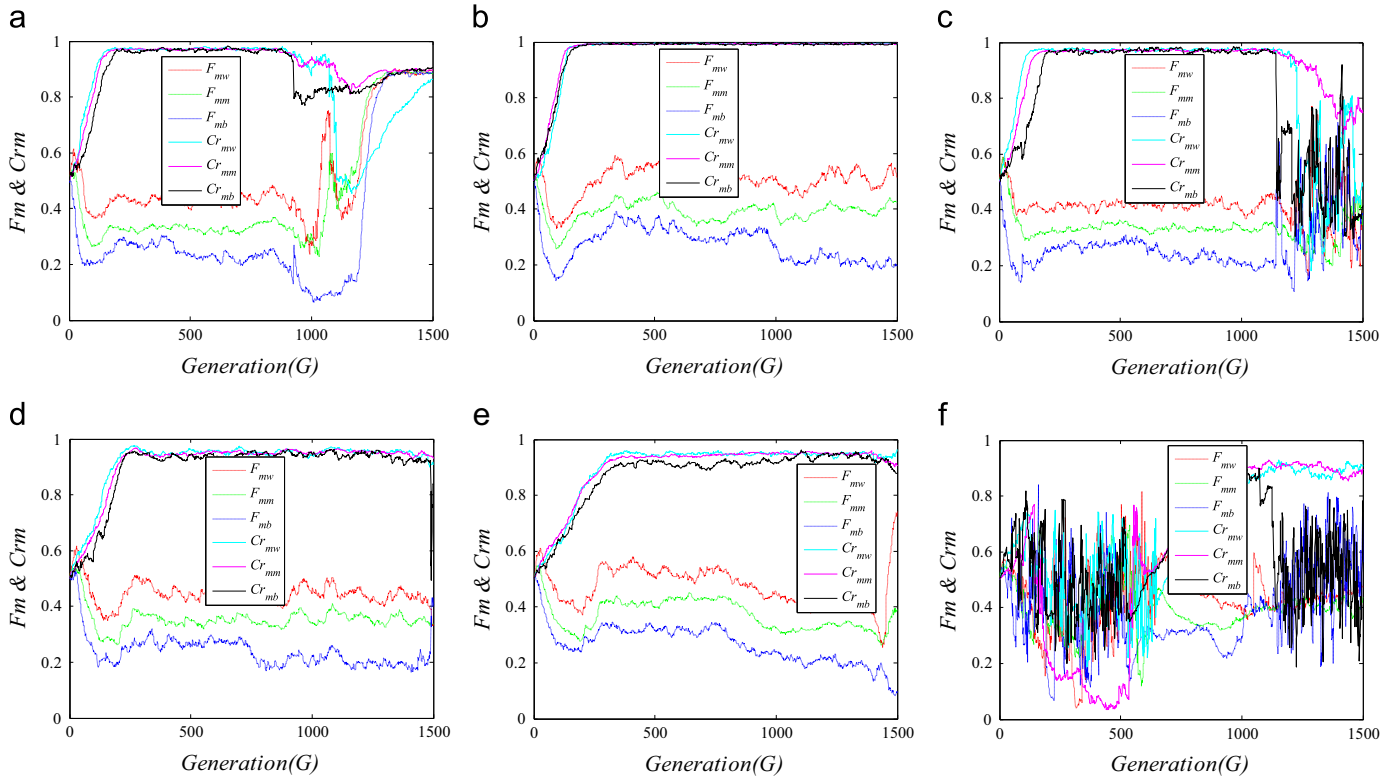

Fig. 5. Evolutionary trend of F_m^G and Cr_m^G of MPADE on (a) *F*2 (b) *F*3 (c) *F*4 (d) *F*5 (e) *F*6 (f) *F*10 with 30*D*.

Table 11
Mean error values of 30 independent runs on *F*2–*F*7, *F*9–*F*14, *F*16 and *F*17 with 30*D* using MPADE, MPDE, JA-MPADE, M-MPADE and MPADE-WRS.

Algorithm	Func						
	*F*2 Mean	*F*3 Mean	*F*4 Mean	*F*5 Mean	*F*6 Mean	*F*7 Mean	*F*9 Mean
MPDE	2.24E−04	1.06E+05	2.11E−02	4.77E−04	2.49E−06	8.22E−03	1.65E+02
JA-MPADE	5.81E−28	6.86E−17	1.54E−23	4.37E−07	3.06E+01	3.78E−03	6.63E−02
M-MPADE	7.68E−28	3.40E+02	1.39E−27	4.51E−05	7.97E−01	7.88E−03	3.81E+00
MPADE-WRS	4.90E−28	8.24E−10	1.21E−27	9.38E−11	1.33E−01	2.14E−03	2.19E−06
MPADE	**3.94E−28**	**4.89E−17**	**5.67E−28**	**2.23E−11**	**1.39E−25**	**1.56E−03**	**0.00E+00**
Algorithm	Func						
	*F*10 Mean	*F*11 Mean	*F*12 Mean	*F*13 Mean	*F*14 Mean	*F*16 Mean	*F*17 Mean
MPDE	1.82E+02	3.95E+01	**1.37E+03**	1.61E+01	1.32E+01	2.24E+02	2.55E+02
JA-MPADE	3.50E+01	1.84E+01	4.94E+03	1.72E+00	**1.18E+01**	6.73E+01	1.26E+02
M-MPDE	3.14E+01	1.63E+01	2.67E+03	2.27E+00	1.22E+01	7.10E+01	8.42E+01
MPADE-WRS	3.33E+01	2.71E+01	2.42E+03	1.79E+00	1.28E+01	6.52E+01	1.61E+02
MPADE	**2.09E+01**	**1.26E+01**	1.90E+03	**1.44E+00**	1.21E+01	**5.05E+01**	**5.27E+01**

performs best on most of test functions and only gets the second ranks on two test functions (*i.e.*, *F*12 and *F*14). These experimental results demonstrate that the parameter adaptation scheme of MPADE is effective as MPADE is much better than MPDE, and also superior to the parameter adaptation scheme proposed in JADE and MDE-*p*BX. Moreover, it is noted that MPDE performs best on *F*12, which indicates that the parameter adaptation scheme cannot always perform best. Furthermore, it is observed from the results of MPADE-WRS that the performance of MPADE will degrade greatly on a majority of benchmark functions without using the replacement strategy. This in turn confirms the advantage of the replacement strategy used in our scheme.

5.3.5. *Sensitivity analysis of some parameters*

The parameters w_1, w_2, w_3 determine the sizes of the inferior, the medium and the superior sub-populations respectively. It can be observed from the previous experiments that MPADE performs better when the values of w_1, w_2, w_3 are set as $w_1 = 1, w_2 = w_3 = 0$ (MPADE_*r*best) or $w_2 = 1, w_1 = w_3 = 0$ (MPADE_*p*best). However, the performance will degrade obviously on all the unimodal and some multimodal functions when they are set to $w_3 = 1, w_1 = w_2 = 0$ (MPADE_*n*best). Thus, the performances of MPADE are actually sensitive to the parameters w_1, w_2, w_3, especially on the unimodal functions. In order to investigate the sensitivities of w_1, w_2, w_3, lots of experiments on some selected functions (*i.e.*, *F*3, *F*4, *F*5, *F*6, *F*9,

Table 12
The combination values of w_1, w_2 and w_3.

$w_1+w_2=1.0$		$w_1+w_2=0.8$		$w_1+w_2=0.6$		$w_1+w_2=0.4$		$w_1+w_2=0.2$		$w_1+w_2=0$	
w_1	w_2	w_1	w_2	w_1	w_2	w_1	w_2	w_1	w_2	w_1	w_2
1	0	0.7	0.1	0.5	0.1	0.3	0.1	0.1	0.1	0	0
0.7	0.3	0.5	0.3	0.3	0.3	0.2	0.2				
0.5	0.5	0.3	0.5	0.1	0.5	0.1	0.3				
0.3	0.7	0.1	0.7								
0	1										

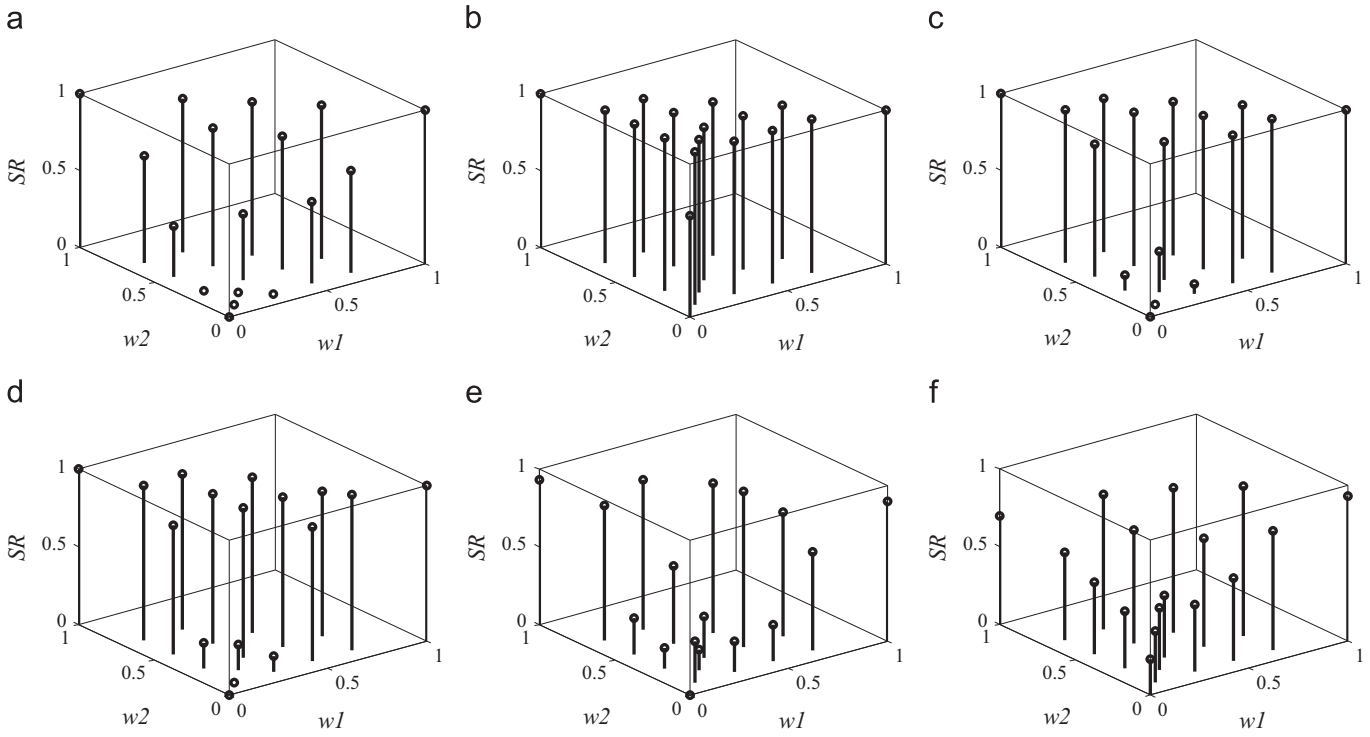

Fig. 6. Success rates of *F*3, *F*4, *F*5, *F*6, *F*9, *F*10 in 30 independent runs with different combination values of w_1, w_2, w_3.

*F*10) are performed using different combinations of w_1, w_2 and w_3. Table 12 gives the combinations of w_1, w_2, w_3 used in our experiments. Each combination value for each test function is run independently by 30 times, and the *SR* value is adopted to assess their performance. The acceptable accuracies of *F*3, *F*4, *F*5, *F*6, *F*9, and *F*10 are set to $1E-15$, $1E-20$, $1E-10$, $1E-20$, $1E-20$ and 30 respectively. The experimental results are depicted in Fig. 6. As observed in Fig. 6, a larger value of w_3 greatly downgrades *SR* in most cases and these results fit our previous results in ADE_*n*best. In general, MPADE can obtain the satisfied results when $w_1+w_2 \approx 1$. Therefore, a proper parameter setting of w_1, w_2, w_3 is suggested with $w_1+w_2 \in [0.8, 1]$.

6. Conclusions and future work

In this paper, we propose a novel adaptive multiple sub-populations based DE algorithm (MPADE), where the three sub-populations are built to take on different search tasks. The main contribution of MPADE is to design the three novel DE strategies: DE/current-to-*r*best/2, DE/current-to-*p*best/2, and DE/current-to-*n*best/2, which are correspondingly conducted on the three sub-populations and responsible for the tasks of either exploration or exploitation. DE/current-to-*r*best/2 and DE/current-to-*n*best/2 respectively focus on exploration and exploitation, while DE/current-to-*p*best/2 can effectively achieve the balance between exploration and exploitation. Moreover, an effective parameter adaptation approach is designed to tune the scaling factor and crossover rate used in the three novel DE strategies and a simple replacement strategy is executed at the end of the selection process to fully exploit the useful information from the trial vectors and target vectors. These proposed approaches greatly improve the optimization performance of DE as validated by the experimental results on the 55 benchmark functions from CEC2005 and CEC2014 competition on real parameter optimization, and 15 real world problems from CEC2011. Simulation results show that MPADE performs much better in unimodal, multimodal functions and real world problems, and competitively in other complex hybrid composition functions, when compared to the five state-of-the-art DE variants (*i.e.*, CoDE, JADE, jDE, EPSDE, SaDE, and SAMODE) and other recently proposed DE variants (*i.e.*, AEPD-JADE, DE-VNS, sinDE and rank-jDE). The effectiveness of the proposed approaches in MPADE is experimentally confirmed and their parameter sensitivity is also analyzed.

Considering the future work, the adaptive mutation strategy and dynamic sub-population sizes will be studied to further improve the performance of MPADE. Moreover, MPADE will be investigated to tackle other optimization problems, such as the

constrained optimization problems and the multi-objective optimization problems.

Acknowledgment

This work is supported by the National Natural Science Foundation of China under Grants 61402294, 61170283 and 61402291, National High-Technology Research and Development Program ("863" Program) of China under Grant 2013AA01A212, Ministry of Education in the New Century Excellent Talents Support Program under Grant NCET-12-0649, Guangdong Natural Science Foundation under Grant S2013040012895, Foundation for Distinguished Young Talents in Higher Education of Guangdong, China under Grant 2013LYM_0076, Major Fundamental Research Project in the Science and Technology Plan of Shenzhen under Grants JCYJ20130329102017840, JCYJ20130329102032059 and JCYJ20140828163633977. The authors are grateful to both the editor and anonymous reviewers for their constructive comments, which greatly improve the quality of this paper.

References

[1] Abdul-Rahman OA, Munetomo M, Akama K. An adaptive parameter binary-real coded genetic algorithm for constraint optimization problems: performance analysis and estimation of optimal control parameters. Inf Sci 2013;233:54–86.
[2] Jin Y, Hao JK, Hamiez JP. A memetic algorithm for the minimum sum coloring problem. Comput Oper Res 2014;43:318–27.
[3] Storn R, Price K. Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim 1997;11(4):341–59.
[4] Qin AK, Huang VL, Suganthan PN. Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Trans Evolut Comput 2009;13(2):398–417.
[5] Brest J, Greiner S, Boskovic B, Mernik M, Zumer V. Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. IEEE Trans Evolut Comput 2006;10(6):646–57.
[6] Zhang J, Sanderson AC. JADE: adaptive differential evolution with optional external archive. IEEE Trans Evolut Comput 2009;13(5):945–58.
[7] Islam SM, Das S, Ghosh S, Roy S, Suganthan PN. An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization. IEEE Trans Syst Man Cybern Part B: Cybern 2012;42 (2):482–500.
[8] Wang Y, Cai Z, Zhang Q. Differential evolution with composite trial vector generation strategies and control parameters. IEEE Trans Evolut Comput 2011;15(1):55–66.
[9] Sha DY, Hsu CY. A new particle swarm optimization for the open shop scheduling problem. Comput Oper Res 2008;35(10):3243–61.
[10] Lin Q, Chen J. A novel micro-population immune multiobjective optimization algorithm. Comput Oper Res 2013;40(6):1590–601.
[11] Kovacevic D, Mladenovic N, Petrovic B, Milosevic P. DE-VNS: Self-adaptive Differential Evolution with crossover neighborhood search for continuous global optimization. Comput Oper Res 2014;52:157–69.
[12] Wang Y, Cai Z. Combining multiobjective optimization with differential evolution to solve constrained optimization problems. IEEE Trans Evolut Comput 2012;16(1):117–34.
[13] Zhao SZ, Suganthan PN, Das S. Self-adaptive differential evolution with multi-trajectory search for large-scale optimization. Soft Comput 2011;15(11):2175–85.
[14] Li K, Zhang Q, Kwong S, Li M, Wang R. Stable matching based selection in evolutionary multiobjective optimization. IEEE Trans Evolut Comput 2014;18 (6):909–23.
[15] Halder U, Das S, Maity D. A cluster-based differential evolution algorithm with external archive for optimization in dynamic environments. IEEE Trans Cybern 2013;43(3):881–97.
[16] Gao Z, Pan Z, Gao J. A new highly efficient differential evolution scheme and its application to waveform inversion. IEEE Geosci Remote Sens Lett 2014;11 (10):1702–6.
[17] Q. Zheng, D. Simon, H. Richter, Z. Gao. Differential particle swarm evolution for robot control tuning. In: Proceedings of the American Control Conference (ACC); 2014. p. 5276–81.
[18] Tenaglia GC, Lebensztajn L. A multiobjective approach of differential evolution optimization applied to electromagnetic problems. IEEE Trans Magn 2014;50 (2):625–8.
[19] Tsai JT, Fang JC, Chou JH. Optimized task scheduling and resource allocation on cloud computing environment using improved differential evolution algorithm. Comput Oper Res 2013;40(12):3045–55.
[20] Gämperle R, Müller SD, Koumoutsakos P. A parameter study for differential evolution. Advances in intelligent systems, fuzzy systems. Evolut Comput 2002;10:293–8.
[21] A. W. Iorio, X. Li. Solving rotated multi-objective optimization problems using differential evolution. In: Proceedings of the AI 2004 advances in artificial intelligence; 2005. p. 861–72.
[22] Mezura-Montes E, Velázquez-Reyes J, Coello Coello CA. A comparative study of differential evolution variants for global optimization. In: Proceedings of the 8th annual conference on genetic and evolutionary computation; 2006. p. 485–92.
[23] Sutton AM, Lunacek M, Whitley LD. Differential evolution and non-separability: using selective pressure to focus search. In: Proceedings of the 9th annual conference on genetic and evolutionary computation; 2007. p. 1428–35.
[24] Yu WJ, Shen M, Chen WN, Zhan ZH, Gong YJ, Lin Y. Differential evolution with two-level parameter adaptation. IEEE Trans Cybern 2014;44(7):1080–99.
[25] Fan HY, Lampinen. J. A trigonometric mutation operation to differential evolution. J Glob Optim 2003;27(1):105–29.
[26] Cai Y, Wang J. Differential evolution with neighborhood and direction information for numerical optimization. IEEE Trans Cybern 2013;43(6):2202–15.
[27] Gong W, Cai Z. Differential evolution with ranking-based mutation operators. IEEE Trans Cybern 2013;43(6):2066–81.
[28] Kaelo P, Ali MM. Differential evolution algorithms using hybrid mutation. Comput Optim Appl 2007;37(2):231–46.
[29] Wang H, Rahnamayan S, Sun H, Omran MG. Gaussian bare-bones differential evolution. IEEE Trans Cybern 2013;43(2):634–47.
[30] R. Mallipeddi, P. N. Suganthan. Differential evolution algorithm with ensemble of parameters and mutation and crossover strategies. In: Swarm, evolutionary, and memetic computing; 2010. p. 71–8.
[31] Mallipeddi R, Suganthan PN, Pan QK, Tasgetiren MF. Differential evolution algorithm with ensemble of parameters and mutation strategies. Appl Soft Comput 2011;11(2):1679–96.
[32] Das S, Suganthan PN. Differential evolution: a survey of the state-of-the-art. IEEE Trans Evolut Comput 2011;15(1):4–31.
[33] Angira R, Santosh A. Optimization of dynamic systems: a trigonometric differential evolution approach. Comput Chem Eng 2007;31(9):1055–63.
[34] Das S, Abraham A, Chakraborty UK, Konar A. Differential evolution using a neighborhood-based mutation operator. IEEE Trans Evolut Comput 2009;13 (3):526–53.
[35] Epitropakis MG, Tasoulis DK, Pavlidis NG, Plagianakos VP, Vrahatis MN. Enhancing differential evolution utilizing proximity based mutation operators. IEEE Trans Evolut Comput 2011;15(1):99–119.
[36] Liu J, Lampinen J. A fuzzy adaptive differential evolution algorithm. Soft Comput 2005;9(6):448–62.
[37] M. G. Omran, A. Salman, A. P. Engelbrecht. Self-adaptive differential evolution. In: Proceedings of the computational intelligence and security; 2005. p. 192–9.
[38] Zou D, Liu H, Gao L, Li. S. A novel modified differential evolution algorithm for constrained optimization problems. Comput Math Appl 2011;61(6):1608–23.
[39] Suganthan PN, Hansen N, Liang JJ, Deb K, Chen YP, Auger A. Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. KanGAL Report, 2005005; 2005.
[40] Das S, Suganthan PN. Problem definitions and evaluation criteria for CEC 2011 competition on testing evolutionary algorithms on real world optimization problems. Singapore: Jadavpur University, India and Nanyang Technological University; 2010 Technical report.
[41] Elsayed, Saber M., Sarker Ruhul A., and Essam Daryl L. Differential evolution with multiple strategies for solving CEC2011 real-world numerical optimization problems. In: Proceedings of the 2011 IEEE Congress on Evolutionary Computation (CEC); 2011. p. 1557–64.
[42] Draa Amer, Bouzoubia Samira, Boukhalfa Imene. A sinusoidal differential evolution algorithm for numerical optimisation. Appl Soft Comput 2015;27:99–126.
[43] Liang JJ, Qu BY, Suganthan PN. Problem definitions and evaluation criteria for CEC 2014 special session and competition on single objective real-parameter numerical optimization. Technical report. Singapore:Nanyang Technological University; China: Zhenzhou University; 2013. [Online].Available at: ⟨http://www.ntu.edu.sg/home/epnsugan/⟩.
[44] Yang Ming, Li Changhe, Cai Zhihua, Guan Jing. Differential evolution with auto-enhanced population diversity. IEEE Trans Cybern 2015;45(2):302–15.
[45] R. Tanabe, A. Fukunaga, Success-history based parameter adaptation for differential evolution. In: Proceedings of the 2013 IEEE Congress on Evolutionary Computation (CEC); 2013. p.71–8.
[46] R. Tanabe, A. Fukunaga, Improving the search performance of SHADE using linear population size reduction. In: Proceedings of the 2014 IEEE Congress on Evolutionary Computation (CEC); 2014. p. 1658–65.
[47] R. Mallipeddi, P. N. Suganthan, Ensemble Differential Evolution Algorithm for CEC2011 problems. In: Proceedings of the 2011 IEEE Congress on Evolutionary Computation (CEC); 2011. p. 1557–64.