Applied Intelligence (2018) 48:3612–3629
https://doi.org/10.1007/s10489-018-1153-y

Differential evolution algorithm with multiple mutation strategies based on roulette wheel selection

Wuwen Qian[1] · Junrui Chai[1] · Zengguang Xu[1] · Ziying Zhang[1]

Published online: 11 April 2018

Abstract
In this paper, we propose a differential evolution (DE) algorithm variant with a combination of multiple mutation strategies based on roulette wheel selection, which we call MMRDE. We first propose a new, reflection-based mutation operation inspired by the reflection operations in the Nelder–Mead method. We design an experiment to compare its performance with seven mutation strategies, and we prove its effectiveness at balancing exploration and exploitation of DE. Although our reflection-based mutation strategy can balance exploration and exploitation of DE, it is still prone to premature convergence or evolutionary stagnation when solving complex multimodal optimization problems. Therefore, we add two basic strategies to help maintain population diversity and increase the robustness. We use roulette wheel selection to arrange mutation strategies based on their success rates for each individual. MMRDE is tested with some improved DE variants based on 28 benchmark functions for real-parameter optimization that have been recommended by the Institute of Electrical and Electronics Engineers CEC2013 special session. Experimental results indicate that the proposed algorithm shows its effectiveness at cooperative work with multiple strategies. It can obtain a good balance between exploration and exploitation. The proposed algorithm can guide the search for a global optimal solution with quick convergence compared with other improved DE variants.

Keywords Differential evolution · Nelder–mead method · New mutation operation · Roulette wheel selection · Multiple mutation strategies · Global optimization

1 Introduction

The differential evolution (DE) algorithm proposed by Price and Storn [1] is a simple but powerful implementation of stochastic search techniques. It has the advantages of simple principles, few controlled parameters, and strong robustness [1–3] for real-valued parameter optimization. Population-based optimization algorithms, including DE, tend to suffer from long computational times because of their evolutionary stochastic nature. This crucial drawback sometimes limits their applications to problems with few or no real-time constraints. Despite this drawback, the DE algorithm has a strong real-number domain-search capability, and the improved DE algorithm is still an important research field whose purpose is to adapt to more complex optimization problems and achieve high-quality solutions.

DE mainly has three control parameters, population size *NP*, scaling factor *F*, and crossover rate *CR*, and many mutation strategies have been proposed. [5, 9, 16, 24] Some mutation strategies are more suitable for global convergence, [34] some are more effective in the local search, [7] and some are more suitable when solving rotated problems. [4] Control parameters of the classical DE are fixed, and a single mutation strategy is used. The performance of DE may vary greatly for different optimization problems using the same control parameters and single offspring generation strategy. [34] However, for a given optimization problem, the most suitable control parameters and the proper mutation strategy may not be the same at different stages of evolution. [34] There are many large-scale optimization problems in the real world. For some problems, we may have to spend a large amount of time to determine the best strategy and tuning parameters

✉ Wuwen Qian
qianwuwen@163.com

[1] State Key Laboratory of Eco-hydraulics in Northwest Arid Region of China (Xi'an University of Technology), Xi'an, 710048, China

[5, 6] to obtain better performance. A DE with outstanding performance is required to achieve a global optimum without considering the characteristics of the problem.

Much attention has been focused on methods to automatically tune parameters and to develop an excellent mutation strategy or ensemble of mutation strategies. This focus has resulted in many improved DE variants, such as SaDE [7] (ensemble of two mutation strategies and self-adapted parameters), jDE [8] (self-adaptive tuning parameters), JADE [9] (to develop an excellent mutation strategy and self-adapted parameters), EPSDE [10] (ensemble of mutation strategies and parameters), CODE [11] (composition of mutation strategies with their own parameters), SHADE [12] (self-adapted parameters), HSDE [13] (hybrid of two mutation strategies), GADE [14] (greedy adapted parameters), MPEDE [15] (ensemble of multiple mutation strategies), and DE/Alopex/1 [16] (to develop an excellent mutation strategy).

The Nelder–Mead method is a traditional direct algorithm for unconstrained nonlinear optimization problems. It has excellent local search capability, but it has strong dependence on the initial solution and it falls easily into a local minimum. Reflection, expansion, contraction, and shrinking are its primary operations. In this paper, inspired by the reflection operations in the Nelder–Mead method, we propose a new, reflection-based mutation operation. For each individual to be mutated, we randomly select four individuals from the current population, calculate the weighted center of the three individuals with better fitness based on their object function values, and use the difference vector of the best and worst individuals as the perturbation vector. We use the center we have calculated as the target individual, thus completing a mutation operation. We compare the reflection-based strategy and seven other mutation strategies in an experiment to prove that the proposed mutation strategy has good exploration and exploitation capability.

Although the reflection-based strategy can balance exploration and exploitation better than other basic mutation strategies, it is still a greedy mutation strategy and is prone to premature convergence when solving complex multimodal optimization problems. We add two basic strategies to generate perturbation vectors to maintain population diversity and to increase robustness. We use a roulette wheel selection strategy to arrange mutation strategies based on their success rates for each individual. We call this variant DE with multiple mutation strategies based on roulette wheel selection (MMRDE). We used 28 benchmark functions to test the performance of MMRDE with some improved DE variants. Experimental results indicate that the proposed algorithm can balance between exploration and exploitation well. The algorithm shows the effectiveness of three mutation strategies in cooperative work. It can guide the search for a global optimal solution with fast convergence compared with other improved DE variants.

This paper has six sections. Section 2 discusses relevant works. Section 3 explains the traditional DE algorithm. We introduce the details of our proposed MMRDE in Section 4. Section 5 discusses the experimental study to test the new reflection-based mutation strategy with other mutation strategies and compares the performance of MMRDE to some improved DE variants. We provide our conclusions in Section 6.

2 Relevant works

The DE algorithm is an excellent evolutionary algorithm (EA) that is simple, has few controlled parameters, and is robust. DE, however, is prone to suffer from premature convergence, evolution stagnation, and long computational times. The performance of DE strongly depends on the setting of control parameters and/or the mutation strategy used.

DE mainly uses the three control parameters *NP*, *F*, and *CR*. *NP* is used to deal with the problem of different dimensionalities, and *F* has greater influence on the convergence speed of DE. [7] The choice of *F* is more flexible. Storn et al. [1] said that a value of *F* in [0.5, 1] is reasonable, a value less than 0.4 or more than 1.0 can sometimes be effective, and a value of 0.6 or 0.5 is often a good choice. As discussed in the literature, [17] an initial *F* value of 0.9 is recommended. According to Qin et al., [7] a value of *F* generated randomly in the range (0, 1] is usually preferred. *CR* mainly affects the adaptability of DE to solve complex problems. A proper value of *CR* may result in good performance in some learning strategies, and the wrong choice may lead to deterioration in the performance of any learning strategy. A good *CR* parameter value usually falls in a small range, and the algorithm always performs well on a complex problem. As discussed in the literature, [1] a large *CR* can accelerate the convergence rate, an initial value of 0.1 is a good choice, and setting *CR* to 0.9 or 1.0 can accelerate convergence. Price and Storn [18] proposed setting *CR* in the range (0, 0.2) for separable problems, and (0.9, 0.1) for multimodal problems. In the previously noted research, we see different conclusions for the tuning of control parameters, but we know that the proper parameters will vary according to the type of problem. Adaptively controlling the parameters within the entire evolution process can enhance the performance of DE. [19]

The control parameter settings of DE can presently be classified into the three categories of deterministic, adaptive, and self-adaptive parameter control. [20] Lampinen [21] proposed a fuzzy adaptive DE in which the values of F and CR are dynamically adapted based on fuzzy logic controllers. Brest et al. [8] proposed a self-adapting control parameter scheme (jDE), which may be the first in a category of literature discussing self-adapting control parameters. Ghosh et al. [22] presented an improved DE (FiADE) algorithm with tuning control parameters based on individual object function values. Leon et al. [14] proposed an adapting DE (GADE) that performs a greedy search to adjust the control parameters. Zhang et al. [9] proposed a self-adaptive control parameter adjustment scheme (JADE) in which F and CR are generated randomly according to Cauchy and normal distributions, respectively.

The chosen mutation strategy has a profound effect on the performance of DE. [23] The idea of designing a new mutation strategy [24] or introducing a technique that automatically selects the classical mutation strategy is used to improve DE performance. A new mutation strategy, DE/current-to-$p$best, with an optional external archive proposed by Zhang et al., [9] is a variant of DE/current-to-best. DE/current-to-$p$best/1 randomly chooses $p\%$ of the population's best individuals instead of the best individual, as in DE/current-to-best/1. A novel mutation strategy called DE/current-to-gr_best/1 [25] is a variant of DE/current-to-best/1 proposed for the binomial crossover of DE. DE/current-to-gr_best/1 randomly chooses $q\%$ of the population as a group and uses the best solutions of the group to perturb the target vector. An improved DE variant named IDE [26] proposes a new triangular mutation strategy to balance global exploration and exploitation of DE. This strategy uses three randomly chosen vectors to form a convex combination vector, which is used as the base vector of the mutation strategy. This triangular mutation strategy uses the difference between the best and worst individuals among the three randomly selected vectors as the disturbance vector. This triangular mutation strategy combines with the basic mutation strategy (DE/rand/1/bin) through a nonlinear decreasing probability. Wang et al. [27] proposed an improved DE variant (IMSaDE) by using an improved mutation strategy that incorporated an elite archive strategy and parameter-adaptation strategy in the basic mutation strategy DE/rand/2. Using different offspring individual generation patterns in an optimization loop can improve the quality of subsequent offspring individuals. Moreover, the optimal individual's relevant information can be passed to the rest of the individuals, which may involve different variation strategies. So, it can also improve the performance of the next generation. The use of different mutation strategies [28] can provide a balance between exploration and exploitation. This also reduces stagnation and premature convergence in DE. Two DE variants with adaptive strategy selection were proposed by Gong et al. [29] to autonomously select the most suitable strategy. Yi et al. [13] proposed an improved DE variant with a hybrid mutation strategy and self-adapting control parameters. The hybrid mutation strategy uses two types of mutation operators and autonomously selects the mutation operators for each individual. Leon et al. [16] proposed a novel mutation strategy called DE/Alopex/1, which calculates the probabilities of move directions based on the fitness value between two individuals, while the movement direction of other mutation strategies is independent of the fitness of selected individuals.

3 The basic DE

DE has proven to be an efficient and effective global algorithm for real-code optimization, and it has been shown to outperform the genetic algorithm. DE is similar to other intelligent optimization algorithms (e.g., genetic algorithms). It also randomly generates a population of *NP* feasible solutions $P_G = \{X_{1,G}, X_{2,G}, \ldots, X_{NP,G}\}$. Any one feasible solution is based on constraint-optimization decision variables in the equation. We call it an individual $X_{i,G} = \left(x_{i,G}^1, x_{i,G}^2, \ldots, x_{i,G}^D\right), i = 1, 2, \ldots, NP$, where G denotes the number of generations and D is the number of individual dimensions. Each dimension of the individual is determined randomly and uniformly in the range $[X_{min}, X_{max}]$, where $X_{\min} = \left(x_{\min}^1, x_{\min}^2, \ldots, x_{\min}^D\right)$ and $X_{\max} = \left(x_{\max}^1, x_{\max}^2, \ldots, x_{\max}^D\right)$. Such a series of random individuals constitute the initial population, which can be expressed as

$$x_i^j = x_{\min}^j + rand\,(0, 1) * \left(x_{\max}^j - x_{\min}^j\right), \quad (1)$$

where *rand(0,1)* denotes a real number that is determined randomly and uniformly between 0 and 1.

3.1 Mutation

DE utilizes the difference vector of parents as its basic variance components. Each difference vector is obtained by subtraction between two different individuals from the parent. The mutation vector is obtained by the addition of the scaled difference vector and different individuals. The DE operator is usually represented by DE/x/y/z, where x represents the perturbed vector (also called the base vector), y represents the number of differential vectors, and z represents the crossover operator. The classical DE algorithm uses DE/rand/1 to generate the mutant vectors,

$$V_{i,G} = X_{a,G} + F \times \left(X_{b,G} - X_{c,G}\right), a \neq b \neq c \neq i, \quad (2)$$

where $X_{a,G}$, $X_{b,G}$, and $X_{c,G}$ are chosen from the current population, and $X_{i,g}$ is the target individual. $X_{a,G}$, $X_{b,G}$, and $X_{c,G}$ are three random individuals. F is a real and constant factor to scale the difference vector.

Some other mutation strategies are used in the literature, as shown in Table 1.

$X_{best,G}$ represents the vector with the best fitness value at generation$G$. $X_{a,G}$, $X_{b,G}$, $X_{c,G}$, $X_{d,G}$, and $X_{e,G}$ are five individuals randomly chosen from the current population. $X_{i,G}$ is the target vector.

3.2 Crossover operation

To improve the diversity of the population, DE introduces the crossover operation so that at least one component of the vector comes from the variation vector. The details are as follows:

$$U_{i,G+1} = \begin{cases} V_{i,G}^{j}, if\ \left(rand^{j}\ (0,1) \le CR\right) or\ (j = j_{rand}) \\ X_{i,G}^{j}, otherwise \end{cases},$$
$$i = 1, 2, \ldots, NP;\ j = 1, 2, \ldots, D, \tag{3}$$

where *rand* ^{j}*(0,1)*$\in$*[0,1]* is the jth evaluation of a uniform random generator number, *CR* is a user-specified crossover probability in the range [0,1], and the index j_{rand}refers to a randomly chosen dimension, which is used to ensure that the trail vector $U_{i,G+1}$ gets at least one element from $V_{i,G} = \begin{cases} U_{i,G+1}, if\ f\left(U_{i,G+1}\right) < f\left(X_{i,G}\right) \\ X_{i,G}, otherwise \end{cases}, i = 1, 2, \ldots, NP$. This cross-operation, called a binomial uniform crossover, is widely used in the literature. When the binomial uniform crossover is used, the strategies in Table 1 are named *DE/rand/1/bin*, *DE/best/1/bin*, *DE/current-to-best/1/bin*, *DE/best/2/bin*, *DE/rand/2/bin*, and*DE/current-to-rand/1/bin*, respectively.

3.3 Selection operation

Good selection strategies can significantly improve the convergence performance of the algorithm. DE uses a greedy selection strategy. The individual generated through mutation and crossover operations is compared with its parent individual. The better-performing individual is selected as one of the next generation of individuals in a population, as follows:

$$X_{i,G+1} = \begin{cases} U_{i,G+1}, if\ f\left(U_{i,G+1}\right) < f\left(X_{i,G}\right) \\ X_{i,G}, otherwise \end{cases},$$
$$i = 1, 2, \ldots, NP. \tag{4}$$

A series of mutations, crossovers, and selection operations can produce the same number of new individuals comprising the next-generation population. Then, the previous generation population continues the operation until termination conditions are reached. The termination conditions are usually met when the maximum number of iterations or the optimal solution of the relative equations is reached.

4 DE with multiple mutation strategies based on roulette wheel selection

In this section, we will introduce the details of the MMRDE algorithm. First, we propose a new reflection-based mutation operation inspired by the reflection operations in the Nelder–Mead method.

4.1 Reflection-based mutation

The Nelder–Mead method is a traditional direct algorithm for unconstrained nonlinear optimization problems. The method first constructs a simplex, which is a polyhedron containing n+ 1 vertices for n-dimensional optimization. After calculating the object function of each vertex, we determine the best point, the reciprocal second-worst point, and the worst point. Through reflection, expansion, contraction, and shrink operations, a better solution can be found and obtain a new simplex by instead of the worst point with the better solution, then an approximate global optimal solution can be obtained after many iterations. The Nelder–Mead method is simple and easy to implement. It has excellent local search capability and does not require the function to be derivable, so it has a wide range of applications. It has strong dependence on the initial solution, however, and easily falls into a local minimum.

Table 1 Expressions of the mutation strategy

Strategy	Name	Expression
1	DE/rand/1	$V_{i,G} = X_{a,G} + F \times \left(X_{b,G} - X_{c,G}\right)$
2	DE/best/1	$V_{i,G} = X_{best,G} + F \times \left(X_{a,G} - X_{b,G}\right)$
3	DE/current-to-best/1	$V_{i,G} = X_{i,G} + F \times \left(X_{best,G} - X_{i,G} + X_{a,G} - X_{b,G}\right)$
4	DE/best/2	$V_{i,G} = X_{best,G} + F \times \left(X_{a,G} - X_{b,G} + X_{c,G} - X_{d,G}\right)$
5	DE/rand/2	$V_{i,G} = X_{a,G} + F \times \left(X_{b,G} - X_{c,G} + X_{d,G} - X_{e,G}\right)$
6	DE/current-to-rand/1	$V_{i,G} = X_{i,G} + F \times \left(X_{c,G} - X_{i,G} + X_{a,G} - X_{b,G}\right)$

The search direction of reflection and expansion operations is consistent. The expansion operation is performed on the premise that the reflected point is best so far. The contraction operation is performed when the reflected point is worse than the reciprocal second-worst point. Compression operations are performed when both reflection and contraction operations are unable to search for a better solution than the reciprocal second-worst point, and their main function is to compress the search space.

After a certain generation of a DE algorithm, the population individual shows the tendency to gather near the optimal solution. [30, 31] Therefore, the whole DE algorithm's optimization process is similar to the compression operation in the Nelder–Mead method. The reflected point is generated with the worst point reflected through the centroid of the remaining n points after removing the worst point.

Motivated by this operation, we designed what we call a reflection-based mutation strategy, which is similar to the reflection operations in the Nelder–Mead method for the DE algorithm. We randomly select four individuals from the current population. We sort them from good to bad according to their object function values, $f(X_{a,G}) < f(X_{b,G}) < f(X_{c,G}) < f(X_{d,G})$. Then, the center $X_{o,G}$ of the first three individuals is generated according to their function values. The trial vectors are generated with the worst point reflected through the best individual. The details are as follows:

$$V_{i,G} = X_{o,G} + F\left(X_{a,G} - X_{d,G}\right), \text{ and} \tag{5}$$

$$\begin{aligned} X_{o,G} &= w_1 X_{a,G} + w_2 X_{b,G} + w_3 X_{c,G} \\ w_1 &= \frac{f(X_{a,G})}{f(X_{a,G})+f(X_{b,G})+f(X_{c,G})} \\ w_2 &= \frac{f(X_{b,G})}{f(X_{a,G})+f(X_{b,G})+f(X_{c,G})} \\ w_2 &= \frac{f(X_{c,G})}{f(X_{a,G})+f(X_{b,G})+f(X_{c,G})} \end{aligned}, \tag{6}$$

where w_1, w_2, and w_3 are the respective weights for $X_{a,G}$, $X_{b,G}$, and $X_{c,G}$; and F is a mutation scaling factor.

The Nelder–Mead method has excellent local search capability. But it has strong dependence on the initial solution and easily falls into a local minimum. Nelder–Mead approximates the optimum solution of a problem with n variables when the object function is continuous and unimodal in the search space. The method, however, is prone to becoming trapped in a basin of attraction of locally optimal solutions and to converge prematurely when solving multimodal optimization problems. The lack of diversity is an important reason for premature convergence of the algorithm. Population diversity and the convergence rate often can be antagonistic. Because the Nelder–Mead method uses all the points except the worst point to calculate the center point and uses a determined evolutionary direction, it can easily lose diversity and converge prematurely. Our reflection-based mutation strategy uses four random individuals, which are a small part of the population when solving high-dimensional problems, to perform mutation operations. When considering the solution of low-dimensional problems when the population is not large, four individuals are not a small number relative to the population, so we allow for the existence of duplicates of these four individuals, but there are at least two individuals. Therefore, this mutation operation can balance exploration and exploitation. Its performance will be confirmed in Section 5.2.

4.2 Assisted mutation strategies

The reflection-based mutation strategy is a greedy strategy. Although this mutation operation can balance exploration and exploitation better than other basic mutation strategies, it is still prone to premature convergence when solving complex multimodal optimization problems. To make it more suitable for solving as many optimization problems as possible, we must increase the robustness of the algorithm. We add two basic strategies to generate perturbation vectors to maintain population diversity. The characteristics of these two mutation strategies are described below:

1. DE/rand/1/bin: DE/rand/1 is a widely used basic mutation strategy for random searches that is always used together with a binomial crossover. [15, 32, 33] It has strong exploration ability and can effectively maintain the diversity of populations. DE/rand/1 is suitable for solving multimodal optimization problems. It converges more slowly, however, when solving single modal problems. [34]
2. DE/current-to-rand/1 without crossover: This mutation strategy uses information of a random individual in the current population to guide the mutation of the current individual. This mutation operation is applied without crossover. It is especially suitable for solving rotation problems because it is rotation-invariant. [11, 15, 35]

4.3 Selection of mutation strategies

Like the original DE algorithm, only one population is used in our algorithm. Therefore, we face the problem of how to arrange the mutation strategy for each individual. In this study, we use a roulette wheel selection strategy [36–38] to arrange mutation strategies for each individual. Roulette strategies are based on the success rate of each strategy, which is the ratio of the number of better offspring individuals generated by one mutation strategy to the total number of individuals using that strategy.

In the initial generation, all mutation strategies have equal success rates. This ensures that each mutation strategy can be used by almost the same number of individuals. As evolution proceeds, some variation strategies may not be appropriate for the evolution of the current generation, so their success rates will be recalculated. The success rate of mutation strategy k (sr_k) is calculated as follows:

$$sr_k = \sum_{i=1}^{NP} sc_{i,k} \big/ N_k \ , sc_{i,k} = \begin{cases} 1 \text{ if } f\left(U_{i,k,G}\right) < f\left(X_{i,k,G}\right) \\ 0 \text{ otherwise} \end{cases}, \tag{7}$$

where N_k is the total number of individuals assigned mutation strategy k; sr_k refers to the success rate of mutation strategy k (it is a real value and varies between 0 and 1); NP is a population size; $U_{i,k,G}$ indicates an individual with an index of i that uses mutation strategy k; and $sc_{i,k}$ is an indicator that denotes whether individual $U_{i,k,G}$ generates better offspring $X_{i,k,G}$. The higher a mutation strategy's success rate, the greater the probability it is chosen. To ensure that the algorithm does not lose some mutation strategies during evolution, we set the same minimum value of the success rate for all mutation strategies. We then calculate the mutation strategy success rate to be below the predetermined minimum, discard the calculated success rate, and replace this rate with the predetermined minimum. We set the minimum success rate of each mutation strategy to 0.08. In addition, to ensure that the success rate of each strategy can be fully calculated, we calculate the success rate every 10 generations.

4.4 Parameter self-adaptive adjustment mechanism

The parameter self-adaptive adjustment mechanism allows an individual to use a scaling factor and crossover factor to generate new individuals, greatly improving convergence performance. The parameter adaptive regulation method we adopt is based on Tanabe et al., [12] which in turn is based on JADE. [9] Two history memories, M_F and M_{CR}, each with H entries, store the mean values of a Cauchy and normal distribution, respectively. F_i and CR_i are scale and crossover factors, respectively, of individuals from the current population. F_i and CR_i are randomly generated according to their respective distributions. The variation factors and crossover factors of an individual are self-adaptively generated as follows:

$$F_i = randc_i\left(M_{F,ri}, 0.1\right), \text{ and} \tag{8}$$

$$CR_i = randn_i\left(M_{CR,ri}, 0.1\right), \tag{9}$$

where $M_{F,ri}$ and $M_{CR,ri}$ are elements that are randomly selected from M_F and M_{CR}, respectively. Both M_F and M_{CR} are initialized to 0.5. To update M_F and M_{CR} expediently, we introduce an index pos to remember the update position, with pos initialized to 1. If $pos > H$, then $pos = 1$, and 1 is added to pos at the end of each generation. F_i is truncated to 1 when $F_i > 1$; when $F_i < 0$, F_i is considered to have an invalid value and it must be regenerated. If $CR_i < 0$, then CR_i is truncated to 0, and if $CR_i > 1$ then CR_i is truncated to 1. $M_{F,pos}$ and $M_{CR,pos}$ are recalculated at the end of each generation as follows:

$$M_{F,pos,G+1} = \begin{cases} mean_{WL}\left(S_F\right) & \text{if } \neq \emptyset \\ M_{F,pos,G} & \text{otherwise} \end{cases}, \tag{10}$$

$$M_{CR,pos,G+1} = \begin{cases} mean_{WA}\left(S_{CR}\right) & \text{if } \neq \emptyset \\ M_{CR,pos,G} & \text{otherwise} \end{cases}, \tag{11}$$

$$mean_{WA}\left(S_{CR}\right) = \sum_{k=1}^{|S_{CR}|} w_k \cdot S_{CR,k}, \tag{12}$$

$$mean_{WL}\left(S_F\right) = \frac{\sum_{k=1}^{|S_F|} w_k \cdot S_{F,k}^2}{\sum_{k=1}^{|S_F|} w_k \cdot S_{F,k}}, \text{ and} \tag{13}$$

$$w_k = \frac{\Delta f_k}{\sum_{k=1}^{|S_{CR}|} \Delta f_k}, \tag{14}$$

where S_F and S_{CR} are sets of all success mutation factors and crossover factors, respectively. They inherit the F_i and CR_i of the successful individuals in each generation. That is, if a trial vector $U_{i,G}$ has a lower object value and then takes this operator, then $F_i \to S_F$, $CR_i \to S_{CR}$. A weight strategy is introduced to control the generation of $M_{F,i}$ and$M_{CR,i}$. This operation effectively alleviates the obvious bias of $M_{CR,i}$ to a small value during the evolution process, according to Peng et al., [39] by introducing an array Δf to record the deviation of the old and new fitness values of the successful individuals, that is, $\Delta f(k) = f_{old}(k) - f_{new}(k)$. The size of the array Δf is equal to the number of successful individuals in the current generation. To aid in understanding the details of MMRDE, Fig. 1 displays its framework.

5 Experimental Setup

This section shows all the experiments we performed for this study. Our algorithms were coded using Fortran 90. The compiler was Intel Visual FORTRAN Composer XE 2013 with VS2010. We ran the algorithms on a PC with 2.83 GHz Intel(R) Core(TM)2 Quad CPU and 4GB RAM on Windows 7. A benchmark set was used to test the performance of the proposed algorithm (MMRDE).

5.1 Introduction of the benchmark set

The benchmark set used in this study consists of 28 benchmark functions from the Institute of Electrical and Electronics Engineers CEC2013 special session on real-parameter optimization. A detailed description of the 28

Fig. 1 The complete pseudo-code of MMRDE

```
Algorithm: MMRDE
NP: population size; D: Dimension; Max_FES: Maximum function evaluation number; RP[:]: Array to memory success rate of mutation strategies with three elements; FEs: Function evaluation counter; iter: Generation; S[:]: Array to memory mutation strategies number with NP entities;
1:  Set MF=0.5, MCR=0.5, RP[1:3]=1/3, iter=0
2:  Randomly generate NP D-dimensional variables individuals according to Eq. 1.
3:  Evaluate the initial population, FES = NP.
4:  While FEs≤Max_FES
5:    iter = iter + 1
6:    Use roulette wheel selection to arrange mutation strategies for each individual S(:)
7:    For i=1 to NP do
8:      Generation control parameters $F_i$ and $CR_i$ according to Eqs. 8 and 9.
9:      Perform the S(i)th mutation strategy and corresponding crossover operations for each individual $X_i$.
10:     if $f(u_{i,G}) \le f(x_{i,G})$ then
11:       $X_{i,G+1}=U_{i,G}$
12:       $S_F \leftarrow F$, $S_{CR} \leftarrow CR$
13:     Else
14:       $X_{i,G+1}=X_{i,G}$
15:     end if
16:   end for
17:   Update $M_F$ and $M_{CR}$ according to Eqs. 10 and 11.
18:   Calculated RP after every 10 generations according to Eq. 7.
19: End While
20: Output the best individual as the final result.
```

benchmark functions can be found in Liang et al. [40] Overall, these 28 benchmark functions can be divided into three classes:

1. Unimodal Functions: $f_1 \sim f_5$
2. Basic Multimodal Functions: $f_6 \sim f_{20}$
3. Composition Functions: $f_{21} \sim f_{28}$

Given theoretical thinking that the error values all converge to 0, to avoid influence of the algorithm on the accuracy of the program language, when the optimal individual reached the precision (10^{-8}), the algorithm was deemed to converge, and it exited the calculation.

5.2 Performance test of DE/Reflection/1

5.3 Experimental settings

To demonstrate the performance of our reflection-based strategy, we compare the performance of the algorithm with reflection-based DE/Reflection/1 and with seven other mutation strategies: DE/Alopex/1, [16] DE/rand/1, DE/best/1, DE/current-to-best/1, DE/best/2, DE/rand/2, and DE/current-to-rand/1. The dimensions of all functions are set to 30.

To ensure fair comparisons, all mutation strategies are combined with the binomial cross strategy, and all adopt the same fixed control parameters: mutation factor F = 0.5, crossover rate CR = 0.5. The population size (NP) of all variants is set to 100. The maximum function evaluations number is *max_FES=30000*D* Each algorithm runs independently 30 times, and the average function value (mean) and standard deviation (Std.) of results of these 30 runs are listed in Table 2. To clearly see the performance of the algorithm in each test function, the best performing algorithm results are set in **bold**. For comparison purposes, the Wilcoxon signed-rank test for pairwise samples is conducted between DE/Reflection/1 and seven other variants. The symbols +, =, and −, respectively, indicate that performance of DE/Reflection/1 is significantly better, not significantly better or worse, or significantly worse than other compared algorithms according to the Wilcoxon signed-rank test at the α = 0.05 level. The three lines at the bottom represent the count results of +, =, and −.

5.4 Comparison of DE/Reflection/1 with other basic DE variants

We can observe from Table 2 that DE/Reflection/1 has the highest number of best results among all the variants, accounting for 32.1% of the test functions. DE/Alopex/1 follows with 28.6%. DE/rand/2 has the worst overall performance because it has the highest probability of achieving the worst result, accounting for 64.3% of the

Table 2 Results of DE/Reflection/1 and the other DE variants

	DE/Reflection/1		DE/Alopex/1		DE/rand/1		DE/best/1	
	Mean (Std.)		Mean (Std.)		Mean (Std.)		Mean (Std.)	
F1	**0.00E+00±(0.00E+00)**	=	**0.00E+00±(0.00E+00)**	=	**0.00E+00±(0.00E+00)**	=	**0.00E+00±(0.00E+00)**	=
F2	3.13E+07± (7.65E+06)	−	1.82E+07± (4.17E+06)	+	1.24E+08± (2.57E+07)	−	6.51E+06± (3.24E+06)	−
F3	1.00E+06± (2.86E+06)	+	9.77E+06± (6.51E+06)	=	**1.20E+05± (3.63E+05)**	+	2.00E+08± (3.01E+08)	+
F4	1.54E+04± (2.22E+03)	+	2.58E+04± (4.02E+03)	+	5.42E+04± (7.39E+03)	−	6.23E+03± (1.50E+03)	−
F5	**0.00E+00±(0.00E+00)**	=	**0.00E+00±(0.00E+00)**	=	0.**00E+00±(0.00E+00)**	=	**0.00E+00±(0.00E+00)**	+
F6	**1.54E+01±(2.09E+00)**	+	1.63E+01± (1.90E+00)	+	1.59E+01± (2.25E+00)	+	4.85E+01± (2.05E+01)	+
F7	4.16E− 01± (8.03E− 01)	+	7.72E+00± (4.09E+00)	+	1.97E+01± (4.81E+00)	+	5.29E+01± (3.80E+01)	+
F8	2.09E+01± (4.94E− 02)	=	2.09E+01± (5.25E− 02)	=	2.10E+01± (4.14E− 02)	=	2.09E+01± (4.71E− 02)	=
F9	3.85E+01± (1.33E+00)	=	3.87E+01± (1.19E+00)	+	3.94E+01± (1.16E+00)	−	**1.48E+01±(2.66E+00)**	−
F10	4.03E− 03± (5.27E− 03)	+	3.06E− 01± (1.39E− 01)	+	2.65E+00± (8.56E− 01)	+	3.64E+00± (4.71E+00)	+
F11	8.76E+01± (7.11E+00)	−	**5.97E+00±(2.36E+00)**	+	9.92E+01± (6.34E+00)	−	3.72E+01± (1.18E+01)	−
F12	1.68E+02± (1.13E+01)	−	1.59E+02± (8.80E+00)	+	1.93E+02± (9.30E+00)	=	**1.55E+02±(4.42E+01)**	=
F13	1.66E+02± (6.07E+00)	−	**1.57E+02±(9.09E+00)**	+	1.89E+02± (1.01E+01)	+	1.71E+02± (3.05E+01)	=
F14	3.76E+03± (3.42E+02)	−	**8.85E+02±(1.43E+02)**	−	3.44E+03± (2.59E+02)	−	1.03E+03± (4.15E+02)	−
F15	7.08E+03± (2.90E+02)	=	7.16E+03± (2.79E+02)	+	7.27E+03± (2.87E+02)	+	7.31E+03± (2.93E+02)	=
F16	2.54E+00± (2.28E− 01)	=	2.42E+00± (2.67E− 01)	=	2.44E+00± (2.92E− 01)	−	2.39E+00± (2.82E− 01)	=
F17	1.19E+02± (6.33E+00)	−	**7.47E+01±(5.96E+00)**	+	1.29E+02± (7.24E+00)	−	8.60E+01± (1.63E+01)	−
F18	1.93E+02± (1.01E+01)	−	**1.83E+02±(6.88E+00)**	+	2.19E+02± (9.34E+00)	+	2.10E+02± (1.30E+01)	=
F19	1.03E+01± (2.50E− 01)	−	9.34E+00± (2.22E− 01)	+	1.29E+01± (1.67E− 01)	−	**7.38E+00± (4.31E− 01)**	−
F20	1.20E+01± (2.50E− 01)	=	1.20E+01± (2.22E− 01)	+	1.26E+01± (1.67E− 01)	=	1.20E+01± (4.31E− 01)	=
F21	2.75E+02± (5.52E+01)	=	3.04E+02± (5.52E+01)	=	**2.70E+02±(4.46E+01)**	=	3.05E+02± (8.02E+01)	=
F22	3.85E+03± (2.32E+02)	−	1.26E+03± (1.30E+03)	+	4.65E+03± (2.45E+02)	−	**9.60E+02±(2.97E+02)**	−
F23	**7.02E+03±(2.23E+02)**	+	7.25E+03± (3.41E+02)	+	7.41E+03± (3.05E+02)	+	7.25E+03± (3.39E+02)	=
F24	**2.01E+02±(3.76E+00)**	+	2.13E+02± (5.43E+00)	+	2.95E+02± (3.07E+00)	+	2.52E+02± (8.91E+00)	+
F25	**2.42E+02±(5.25E+00)**	+	2.57E+02± (1.41E+01)	+	2.96E+02± (8.51E+00)	+	2.65E+02± (7.01E+00)	+
F26	2.28E+02± (4.34E+01)	−	2.11E+02± (3.03E+01)	=	2.10E+02± (2.03E+00)	+	2.83E+02± (6.77E+01)	−
F27	3.89E+02± (2.37E+02)	+	4.63E+02± (2.48E+02)	+	1.28E+03± (2.78E+01)	+	7.57E+02± (6.25E+01)	+
F28	**3.00E+02± (0.00E+00)**	=	**3.00E+02± (0.00E+00)**	=	**3.00E+02± (0.00E+00)**	=	3.17E+02± (9.13E+01)	+
Ave. Rank:	3.29		2.93		5.11		4.07	
=		9		8		7		10
+		9		19		12		9
−		1		1		9		9

Table 2 (continued)

	DE/current-to-best/1 Mean (Std.)		DE/best/2 Mean (Std.)		DE/rand/2 Mean (Std.)		DE/current-to-rand/1 Mean (Std.)
F1	**0.00E+00± (0.00E+00)**	=	**0.00E+00±(0.00E+00)**	=	0.00E+00± (0.00E+00)	=	9.09E-08± (4.89E-07)
F2	**3.31E+06±(1.31E+ 06)**	+	4.82E+07± (1.87E+07)	+	1.91E+08± (3.18E+07)	−	2.12E+07± (3.71E+06)
F3	3.85E+07± (4.85E+07)	+	5.44E+07± (8.66E+07)	+	1.73E+09± (3.94E+08)	+	7.05E+06± (5.17E+06)
F4	**5.22E+03±(1.11E+03)**	+	2.85E+04± (3.56E+03)	+	7.13E+04± (1.01E+04)	+	2.26E+04± (3.13E+03)
F5	1.51E+00± (2.06E+00)	=	**0.00E+00±(0.00E+00)**	+	4.40E− 06± (1.25E− 06)	+	1.01E+01± (1.17E+01)
F6	7.39E+01± (2.03E+01)	+	1.96E+01± (4.84E+00)	+	2.53E+01± (5.92E+00)	+	5.54E+01± (1.75E+01)
F7	2.06E+01± (1.13E+01)	+	4.25E+01± (1.64E+01)	+	7.75E+01± (5.97E+00)	+	1.76E+00± (1.18E+00)
F8	2.10E+01± (4.59E− 02)	=	2.09E+01± (5.38E− 02)	=	**2.09E+01±(5.50E− 02)**	=	2.09E+01± (6.00E− 02)
F9	2.83E+01± (9.46E+00)	=	3.71E+01± (3.09E+00)	+	3.93E+01± (1.01E+00)	−	3.69E+01± (1.12E+00)
F10	1.25E+01± (1.13E+01)	+	1.07E− 01± (2.44E− 01)	+	4.09E+02± (8.83E+01)	+	4.41E+00± (2.14E+00)
F11	1.89E+01± (9.52E+00)	+	1.10E+02± (8.79E+00)	+	1.28E+02± (7.67E+00)	−	7.72E+01± (5.87E+00)
F12	1.68E+02± (1.36E+01)	+	1.99E+02± (1.29E+01)	+	2.32E+02± (7.42E+00)	=	1.65E+02± (9.28E+00)
F13	1.68E+02± (1.25E+01)	+	1.99E+02± (1.29E+01)	+	2.29E+02± (1.10E+01)	=	1.68E+02± (9.84E+00)
F14	3.49E+03± (4.34E+02)	−	3.53E+03± (3.38E+02)	+	4.30E+03± (1.81E+02)	+	4.58E+03± (2.01E+02)
F15	**7.06E+03±(3.28E+02)**	+	7.34E+03±(3.30E+02)	+	7.40E+03±(2.26E+02)	=	7.18E+03±(1.49E+02)
F16	2.50E+00± (3.26E− 01)	=	2.41E+00± (2.98E− 01)	=	2.51E+00± (2.87E− 01)	−	**2.34E+00±(3.06E−01)**
F17	9.25E+01± (9.37E+00)	+	1.40E+02± (8.05E+00)	+	1.66E+02± (8.82E+00)	−	1.02E+02± (6.02E+00)
F18	1.96E+02± (1.41E+01)	+	2.23E+02± (1.58E+01)	+	2.53E+02± (1.19E+01)	=	1.91E+02± (6.92E+00)
F19	9.56E+00± (4.08E− 01)	+	1.30E+01± (3.06E− 01)	+	1.56E+01± (2.47E− 01)	=	1.07E+01± (2.26E− 01)
F20	**1.20E+01±(4.08E− 01)**	+	1.26E+01± (3.06E− 01)	+	1.29E+01± (2.47E− 01)	+	1.24E+01± (2.26E− 01)
F21	3.03E+02±(8.25E+01)	=	2.82E+02± (7.81E+01)	=	2.72E+02± (3.64E+01)	+	3.16E+02± (5.13E+01)
F22	3.35E+03± (5.02E+02)	+	4.21E+03± (6.04E+02)	+	5.26E+03± (3.34E+02)	+	4.35E+03± (3.10E+02)
F23	7.09E+03± (3.56E+02)	+	7.35E+03± (3.13E+02)	+	7.62E+03± (2.12E+02)	+	7.34E+03± (2.49E+02)
F24	2.27E+02± (1.14E+01)	+	2.50E+02± (1.41E+01)	+	3.02E+02± (2.36E+00)	+	2.01E+02± (1.50E+00)
F25	2.54E+02± (7.05E+00)	+	2.56E+02± (7.60E+00)	+	3.06E+02± (3.31E+00)	+	2.95E+02± (7.28E+00)
F26	2.17E+02± (4.24E+01)	+	2.69E+02± (8.46E+01)	=	2.21E+02± (8.91E+00)	−	**2.01E+02±(2.35E− 01)**
F27	5.27E+02± (1.01E+02)	+	1.05E+03± (1.80E+02)	+	1.30E+03± (2.75E+01)	=	**3.18E+02±(1.64E+01)**
F28	3.74E+02± (2.74E+02)	=	3.37E+02± (1.98E+02)	+	3.00E+02± (2.51E− 04)	+	3.09E+02± (1.01E+01)
Ave. Rank:	3.96		5.46		6.86		4.32
=		7		5		8	
+		12		23		14	
−		1		0		6	

Table 3 Comparison of DE/Reflection/1 with other variants

	Unimodal functions			Basic multimodal functions			Composition functions		
	Better than	Equal to	Worse than	Better than	Equal to	Worse than	Better than	Equal to	Worse than
DE/Alopex/1	2	2	1	3	5	7	4	2	2
DE/rand/1	2	3	0	12	2	1	5	3	0
DE/best/1	1	2	2	6	3	6	5	2	1
DE/current-to-best/1	2	1	2	3	7	5	4	2	2
DE/best/2	3	2	0	11	3	1	6	2	0
DE/rand/2	4	1	0	13	2	0	6	2	0
DE/current-to-rand/1	3	1	1	5	6	4	6	1	1

functions. DE/Reflection/1 mostly achieves the best result on composition functions, so it is more suitable for solving this type of function. DE/Alopex/1 mostly gets the best result on basic multimodal functions. DE/current-to-best/1 is more suitable for solving unimodal functions.

The bottom three lines of Table 2 represent the results of pairwise comparisons between DE/1Reflection/1 and other algorithms. DE/Reflection/1 is better at more functions than other mutation strategies, in addition to DE/Alopex/1 and DE/current-to-best/1. DE/Reflection/1 achieves similar performance to DE/Alopex/1 and DE/current-to-best/1 Table 3 shows more detailed results. From this we see that DE/Reflection/1 performs well for unimodal functions and only slightly worse than DE/best/1. DE/Alopex/1 performs best when calculating basic multimodal functions, and DE/Reflection/1 achieves similar performance to DE/current-to-best/1 and is only worse than DE/Alopex/1. DE/Reflection/1 performs best when solving composition functions. With an average ranking of 3.29, DE/Reflection/1was the second-best algorithm

In Table 4, the Wilcoxon test is applied to all functions. DE/Reflection/1 is significantly better than DE/rand/1, DE/best/2, and DE/rand/2. It, however, is not significantly better or worse than DE/Alopex/1, DE/best/1, DE/current-to-best/1, and DE/current-to-rand/1.

Table 4 The Wilcoxon test between MMRDE and other DE variants using the results from all the functions (significance of 0.05)

	$R+$	$R-$	Asymptotic P-value	Significant?
DE/Alopex/1	195.5	182.5	0.866446	No
DE/rand/1	302.5	75.5	0.006165	Yes
DE/best/1	233.5	172.5	0.480242	No
DE/current-to-best/1	198.0	180.0	0.819462	No
DE/best/2	371.5	34.5	0.000119	Yes
DE/rand/2	355.0	23.0	0.000063	Yes
DE/current-to-rand/1	277.0	129.0	0.089797	No

To compare the convergence speed of each mutation strategy, we choose a set of results from 30 independent runs to draw the convergence process curve. Figure 2 shows the convergence curve of DE/Reflection/1 and other DE variants on some functions from the benchmark set. From this we see that the convergence speed of DE/Reflection/1 is one of the fastest among all the mutation strategies in functions F1, F10, F24, and F25. In Fig. 2a, DE/Reflection/1 is seen to have good local search ability. In Fig. 2g and h, the reflection-variation strategy seems to effectively maintain the population diversity.

5.5 Performance test of MMRDE

5.6 Experimental settings

To compare the overall performance of the DE algorithm with multiple mutation strategies based on roulette wheel selection (MMRDE), we conducted direct and indirect comparisons for all functions on the benchmark set. The four DE variants SHADE, [12] CODE, [11] EPSDE, [10] and JADE [9] are used to perform direct performance comparisons with MMRDE. Their data come directly from the literature [12] . We then compare the performance of MMRDE, HSDE, JADE, and SHADE by our experiment. All the algorithms we used for comparisons were coded using Fortran 90. To establish a consistent test environment, we coded the JADE and HSDE algorithms in FORTRAN based on the literature [9, 13]. The SHADE code was translated from the original C++ program (found in the CEC Shared Documents[1]) to Fortran 90.

We did not specifically tune the parameters of other algorithms for our experiment. To fairly compare the performance of the algorithms, the population size of each algorithm was set to 100 for all dimensions. Each algorithm ran 30 times independently. The dimension of all 28 test

[1] http://www.ntu.edu.sg/home/EPNSugan/index_files/CEC2013/CEC2013.htm

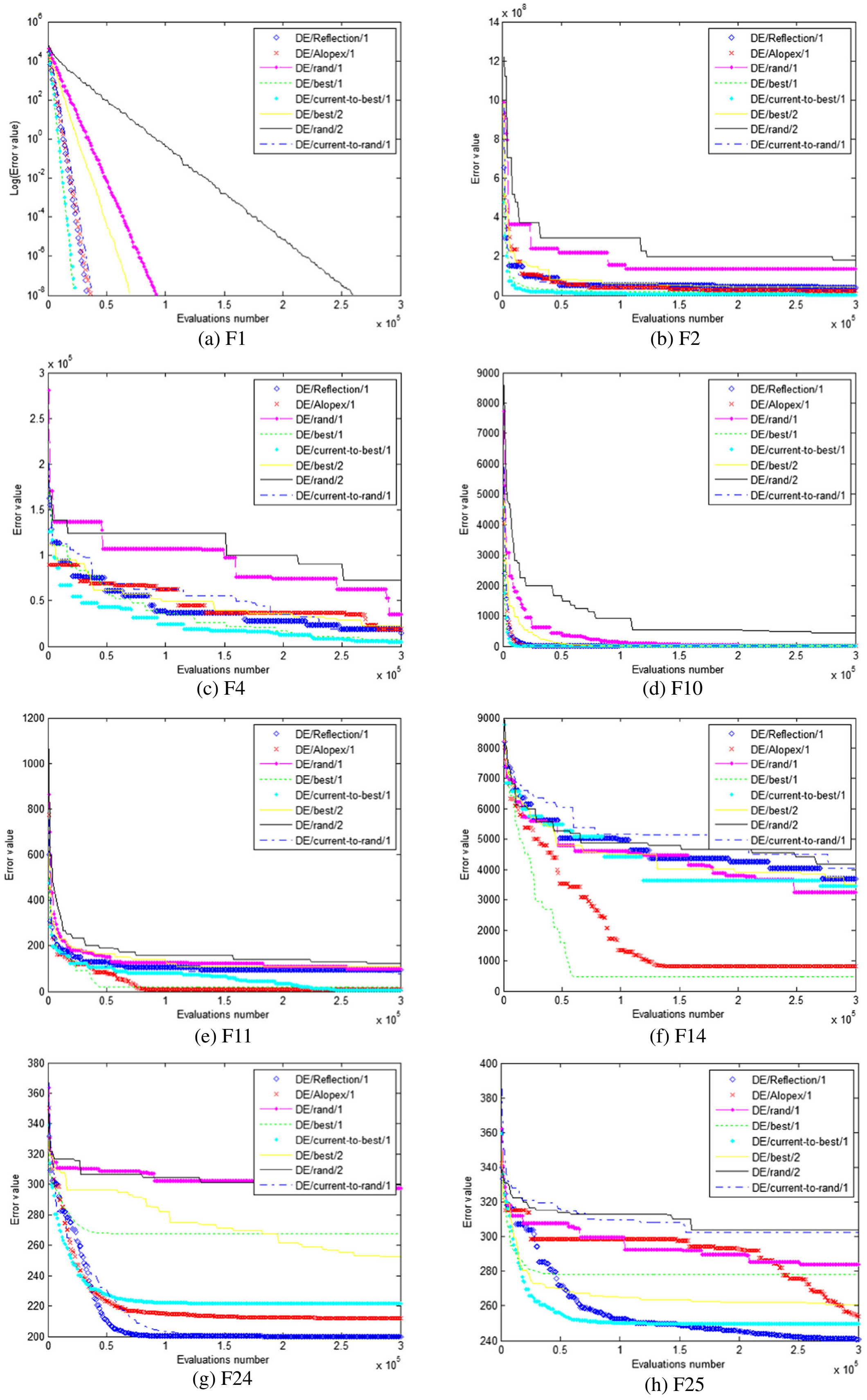


Fig. 2 Comparison of the convergence speed of DE/Reflection/1 and other DE variants

functions in the benchmark set was set to 30. The search space was [−100,100] for all dimensions. The maximum functions number was *max_FES=10000*D*.

The average value (mean) and standard deviation (Std.) of 30 independent calculation results for each function are listed in Table 5 for indirect comparisons and in Table 6 for self-comparisons. To clearly see which algorithm performs best on each test function, the best outcomes among all algorithms are shown in boldface. For comparison, the Wilcoxon signed-rank test is conducted between the state-of-the-art DE variants and MMRDE in Table 6 using KEEL, [41] where +, =, and −, respectively, indicate that performance of our algorithm is significantly better, not significantly better or worse, or significantly worse than other compared state-of-the-art DE algorithms according to the Wilcoxon signed-rank test at the $\alpha = 0.05$ level. The three lines at the bottom of Table 6 represent the count results of +, =, and −, and Table 7 shows more detailed results. At the bottom of Table 6, the average ranks and summation of relative errors between all algorithms are shown.

5.7 Comparison of MMRDE with other DE variants

Table 5 shows the experimental results of indirect comparison with other state-of-the-art DE variants when solving the 28 benchmark functions. The experimental results obtained by running each state-of-the-art DE algorithm 30 times on each function in the benchmark set are shown in Table 6. From the results in Tables 5 and 6, we can summarize the experimental results as follows.

1. Unimodal Functions $f_1 \sim f_5$: MMRDE performs better than HSDE and JADE, because MMRDE obtains significantly better results in more functions. MMRDE has similar performance to SHADE. All the competing

Table 5 Indirect comparison among SHADE, CoDE, EPSDE, JADE, and MMRDE for all functions in CEC2013

	SHADE Mean± (Std.)	CoDE Mean± (Std.)	EPSDE Mean± (Std.)	JADE Mean± (Std.)	MMRDE Mean± (Std.)
F1	**0.00E+00±(0.00E+00)**	**0.00E+00±(0.00E+00)**	**0.00E+00±(0.00E+00)**	**0.00E+00± (0.00E+00)**	**0.00E+00±(0.00E+00)**
F2	9.00E+03± (7.47E+03)	9.78E+04± (4.81E+04)	1.37E+06± (5.23E+06)	**7.67E+03± (5.66E+03)**	1.21E+05± (7.19E+04)
F3	**4.02E+01±(2.13E+02)**	1.08E+06± (3.03E+06)	1.75E+08± (5.39E+08)	4.71E+05± (2.35E+06)	1.55E+03± (6.31E+03)
F4	1.92E-04± (3.01E-04)	8.18E-02± (1.09E-01)	8.08E+03± (2.56E+04)	6.09E+03± (1.33E+04)	3.92E-04± (9.30E-04)
F5	**0.00E+00± (0.00E+00)**	**0.00E+00±(0.00E+00)**	**0.00E+00±(0.00E+00)**	**0.00E+00± (0.00E+00)**	**0.00E+00±(0.00E+00)**
F6	**5.96E-01± (3.73E+00)**	4.16E+00± (9.00E+00)	9.27E+00± (1.33E+00)	2.07E+00± (7.17E+00)	7.39E-01± (1.07E+00)
F7	4.60E+00± (5.39E+00)	9.32E+00± (6.34E+00)	5.88E+01± (4.29E+01)	3.16E+00± (4.13E+00)	**1.96E+00±(1.89E+00)**
F8	2.07E+01± (1.76E-01)	2.08E+01± (1.18E-01)	2.09E+01± (5.32E-02)	2.09E+01± (4.93E-02)	**2.06E+01± (1.44E-01)**
F9	2.75E+01± (1.77E+00)	**1.45E+01±(2.90E+00)**	3.50E+01± (4.21E+00)	2.65E+01± (1.96E+00)	1.68E+01± (4.99E+00)
F10	7.69E-02± (3.58E-02)	2.71E-02± (1.50E-02)	1.02E-01± (5.65E-02)	4.04E-02± (2.37E-02)	5.65E-02± (2.99E-02)
F11	**0.00E+00±(0.00E+00)**	**0.00E+00±(0.00E+00)**	1.95E-02± (1.39E-01)	**0.00E+00±(0.00E+ 00)**	**0.00E+00± (0.00E+00)**
F12	2.30E+01± (3.73E+00)	3.98E+01± (1.21E+01)	4.94E+01± (9.28E+00)	2.29E+01± (5.45E+00)	**2.11E+01± (6.49E+00)**
F13	5.03E+01± (1.34E+01)	8.04E+01± (2.74E+01)	7.68E+01± (1.72E+01)	**4.67E+01± (1.37E+01)**	5.08E+01±(2.06E+01)
F14	3.18E-02± (2.33E-02)	3.60E+00± (4.09E+00)	3.99E-01± (6.00E-01)	2.86E-02± (2.53E-02)	1.82E+00± (8.75E-01)
F15	**3.22E+03±(2.64E+02)**	3.36E+03± (5.31E+02)	6.75E+03±(7.60E+02)	3.24E+03± (3.17E+02)	3.26E+03± (6.54E+02)
F16	9.13E-01± (1.85E-01)	3.38E-01± (2.03E-01)	2.48E+00± (2.88E-01)	1.84E+00± (6.27E-01)	1.41E-01± (6.94E-02)
F17	**3.04E+01±(3.83E-14)**	**3.04E+01± (1.17E-02)**	**3.04E+01±(2.51E-02)**	**3.04E+01± (1.95E-14)**	**3.04E+01±(2.09E-06)**
F18	7.25E+01± (5.58E+00)	6.69E+01± (9.23E+00)	1.37E+02± (1.12E+01)	7.76E+01± (5.91E+00)	**5.38E+01± (1.65E+01)**
F19	**1.36E+00± (1.20E-01)**	1.61E+00± (3.58E-01)	1.84E+00± (2.00E-01)	1.44E+00± (8.71E-02)	1.74E+00± (4.98E-01)
F20	1.05E+01± (6.04E-01)	1.06E+01± (6.69E-01)	1.30E+01± (6.33E-01)	1.04E+01± (5.82E-01)	**1.00E+01± (4.98E-01)**
F21	3.09E+02± (5.65E+01)	**3.02E+02±(9.02E+01)**	3.05E+02± (8.06E+01)	3.04E+02± (6.68E+01)	3.05E+02± (8.02E+01)
F22	9.81E+01± (2.52E+01)	1.17E+02± (9.96E+00)	3.09E+02± (1.12E+02)	**9.39E+01±(3.08E+01)**	1.12E+02± (4.06E+00)
F23	3.51E+03± (4.11E+02)	3.56E+03± (6.12E+02)	6.74E+03± (8.20E+02)	3.36E+03± (4.01E+02)	**3.12E+03± (5.58E+02)**
F24	**2.05E+02± (5.29E+00)**	2.21E+02± (9.28E+00)	2.91E+02± (7.08E+00)	2.17E+02± (1.57E+01)	2.08E+02± (4.35E+00)
F25	2.59E+02± (1.96E+01)	2.57E+02± (6.55E+00)	2.99E+02± (3.29E+00)	2.74E+02± (1.06E+01)	**2.48E+02± (6.16E+00)**
F26	**2.02E+02± (1.48E+01)**	2.18E+02± (4.48E+01)	3.56E+02± (6.49E+01)	2.15E+02± (4.11E+01)	2.04E+02± (1.94E+01)
F27	3.88E+02± (1.09E+02)	6.20E+02± (1.01E+02)	1.21E+03± (7.42E+01)	6.70E+02± (2.40E+02)	**3.54E+02± (2.41E+01)**
F28	**3.00E+02± (0.00E+00)**	**3.00E+02± (0.00E+00)**	**3.00E+02±(0.00E+00)**	**3.00E+02± (0.00E+00)**	**3.00E+02±(0.00E+00)**

Table 6 Comparison among HSDE, JADE, SHADE, and MMRDE for all functions in CEC2013

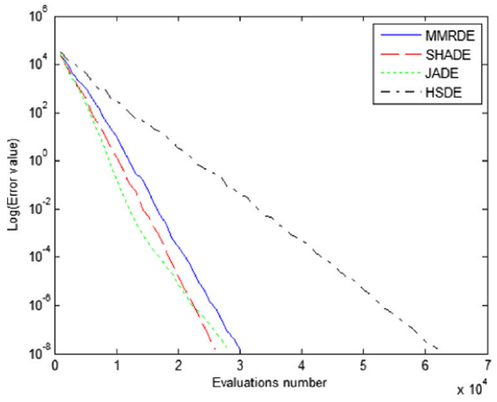
	SHADE		JADE		HSDE		MMRDE
	Mean± (Std.)		Mean± (Std.)		Mean± (Std.)		Mean± (Std.)
F1	**0.00E+00±(0.00E+00)**	=	**0.00E+00± (0.00E+00)**	=	**0.00E+00± (0.00E+00)**	=	**0.00E+00± (0.00E+00)**
F2	**1.88E+04± (1.36E+04)**	−	1.55E+04± (8.07E+03)	−	3.60E+05± (2.14E+05)	+	1.21E+05± (7.19E+04)
F3	8.23E+05± (2.01E+06)	=	1.80E+06± (4.45E+06)	+	2.34E+05± (1.18E+06)	+	**1.55E+03± (6.31E+03)**
F4	2.29E-01± (2.21E-01)	+	3.06E-01± (4.48E-01)	+	1.25E+02± (5.08E+01)	+	3.92E-04± (9.30E-04)
F5	**0.00E+00± (0.00E+00)**	=	**0.00E+00± (0.00E+00)**	=	**0.00E+00± (0.00E+00)**	=	**0.00E+00± (0.00E+00)**
F6	2.25E-03± (8.27E-03)	−	8.80E-01± (4.74E+00)	−	3.16E+00± (6.29E+00)	+	7.39E-01± (1.07E+00)
F7	2.75E+00± (1.95E+00)	=	1.19E+01± (5.07E+00)	+	7.83E+00± (9.81E+00)	+	**1.96E+00± (1.89E+00)**
F8	2.07E+01± (1.37E-01)	+	2.08E+01± (1.73E-01)	+	2.09E+01± (6.29E-02)	+	**2.06E+01±(1.44E-01)**
F9	2.72E+01± (3.57E+00)	+	**1.64E+01± (4.08E+00)**	=	3.30E+01± (1.61E+00)	+	1.68E+01± (4.99E+00)
F10	8.12E-02± (3.89E-02)	+	5.34E-02± (4.08E-02)	=	1.67E-02± (9.05E-03)	−	5.65E-02± (2.99E-02)
F11	**0.00E+00± (0.00E+00)**	=	1.59E+01± (4.21E+00)	+	1.79E+01± (1.64E+00)	+	**0.00E+00± (0.00E+00)**
F12	2.22E+01± (5.25E+00)	=	2.25E+01± (7.04E+00)	=	1.33E+02± (1.42E+01)	+	**2.11E+01± (6.49E+00)**
F13	**4.83E+01± (1.14E+01)**	=	5.57E+01± (1.97E+01)	=	1.52E+02± (1.16E+01)	+	5.08E+01± (2.06E+01)
F14	2.45E-02± (2.21E-02)	−	9.76E+02± (2.59E+02)	+	1.54E+03± (1.50E+02)	+	1.82E+00± (8.75E-01)
F15	**3.21E +03± (3.22E+02)**	=	3.35E+03± (8.51E+02)	=	6.76E+03± (3.12E+02)	+	3.26E+03± (6.54E+02)
F16	9.74E-01± (5.31E-01)	+	3.74E-01± (2.16E-01)	+	2.39E+00± (3.31E-01)	+	1.41E-01± (6.94E-02)
F17	**3.04E+01± (1.42E-14)**	=	4.45E+01± (6.79E+00)	+	6.16E+01± (2.54E+00)	+	**3.04E+01± (2.09E-06)**
F18	7.09E+01± (5.29E+00)	+	5.70E+01± (2.28E+01)	=	2.01E+02± (8.99E+00)	+	**5.38E+01± (1.65E+01)**
F19	**1.52E+00± (4.34E-01)**	−	2.43E+00± (6.82E-01)	=	5.40E+00± (3.59E-01)	=	1.74E+00± (4.98E-01)
F20	1.02E+01± (4.34E-01)	=	1.05E+01± (6.82E-01)	+	1.22E+01± (3.59E-01)	+	**1.00E+01± (4.98E-01)**
F21	**2.81E+02± (5.17E+01)**	=	2.99E+02± (6.89E+01)	=	3.08E+02± (9.18E+01)	=	3.05E+02± (8.02E+01)
F22	**1.01E+02±(1.92E+01)**	−	6.52E+02± (5.32E+02)	+	1.96E+03± (2.91E+02)	+	1.12E+02± (4.06E+00)
F23	3.62E+03± (3.94E+02)	+	3.24E+03± (6.71E+02)	=	6.59E+03± (3.70E+02)	+	**3.12E+03± (5.58E+02)**
F24	**2.06E+02± (4.10E+00)**	=	2.25E+02± (6.93E+00)	+	2.68E+02± (1.82E+01)	+	2.08E+02± (4.35E+00)
F25	2.50E+02± (5.51E+00)	=	2.54E+02± (5.65E+00)	+	2.57E+02± (1.55E+01)	+	**2.48E+02± (6.16E+00)**
F26	2.11E+02± (3.29E+01)	=	2.17E+02± (4.24E+01)	=	2.30E+02± (6.69E+01)	+	**2.04E+02± (1.94E+01)**
F27	**3.52E+02± (3.07E+01)**	=	6.07E+02± (6.91E+01)	+	1.09E+03± (8.57E+01)	+	3.54E+02± (2.41E+01)
F28	**3.00E+02± (0.00E+00)**	=	**3.00E+02± (0.00E+00)**	=	**3.00E+02± (0.00E+00)**	=	**3.00E+02± (0.00E+00)**
Average rank	1.93		2.70		3.63		1.75
Sum. rel. error	12.38		15.63		23.99		11.34
+		7		13		22	
=		16		13		5	
−		5		2		1	

Table 7 Comparison of MMRDE against the other DE variants

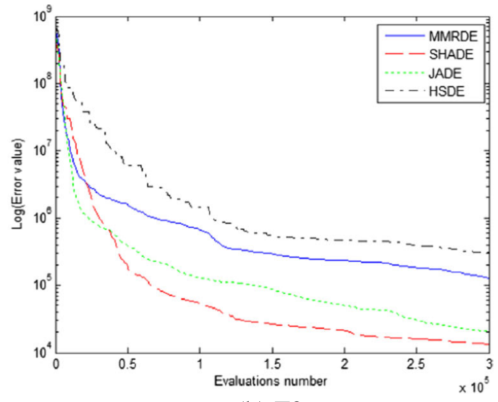
	Unimodal functions			Basic multimodal functions			Composition functions		
	Better than	Equal to	Worse than	Better than	Equal to	Worse than	Better than	Equal to	Worse than
SHADE	1	3	1	5	7	3	1	6	1
JADE	2	2	1	7	7	1	4	4	0
HSDE	3	2	0	13	1	1	6	2	0

Table 8 The Wilcoxon text between MMRDE and other DE variants using the result from all the functions (significance of 0.05)

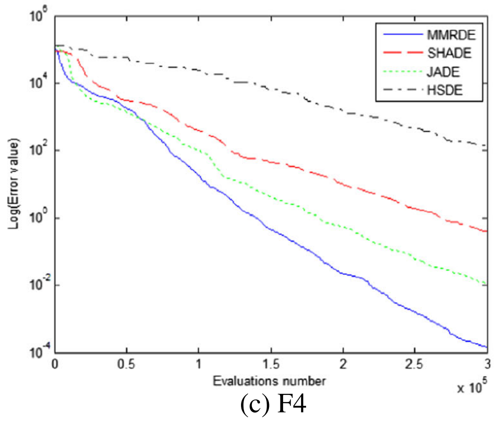
	$R+$	$R-$	Asymptotic P-value	Significance?
SHADE	211.0	195.0	0.846523	No
JADE	324.5	53.5	0.001085	Yes
HSDE	373.5	4.5	0.000009	Yes



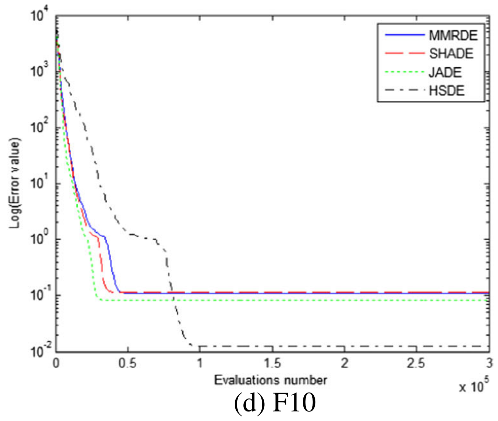
(a) F1



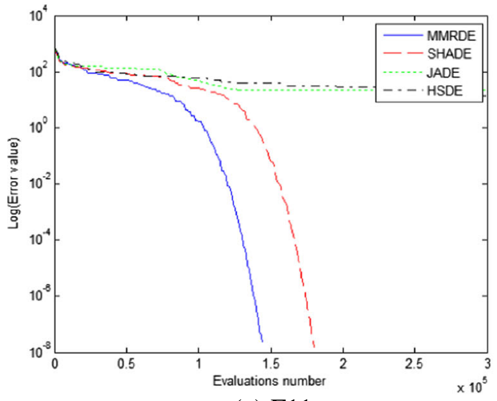
(b) F2



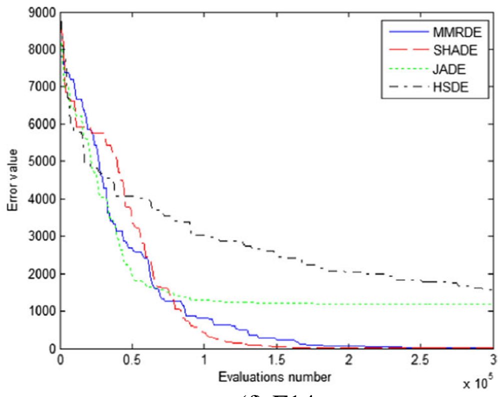
(c) F4



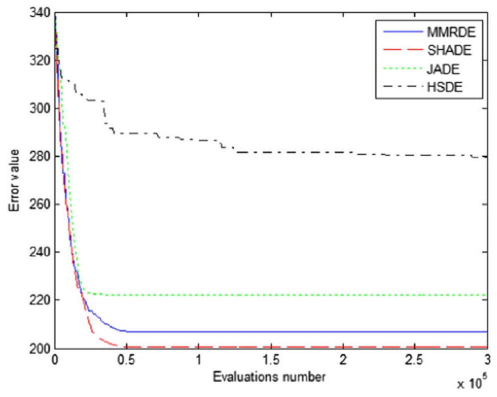
(d) F10



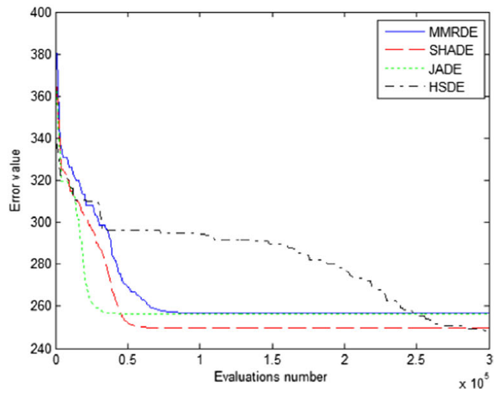
(e) F11



(f) F14

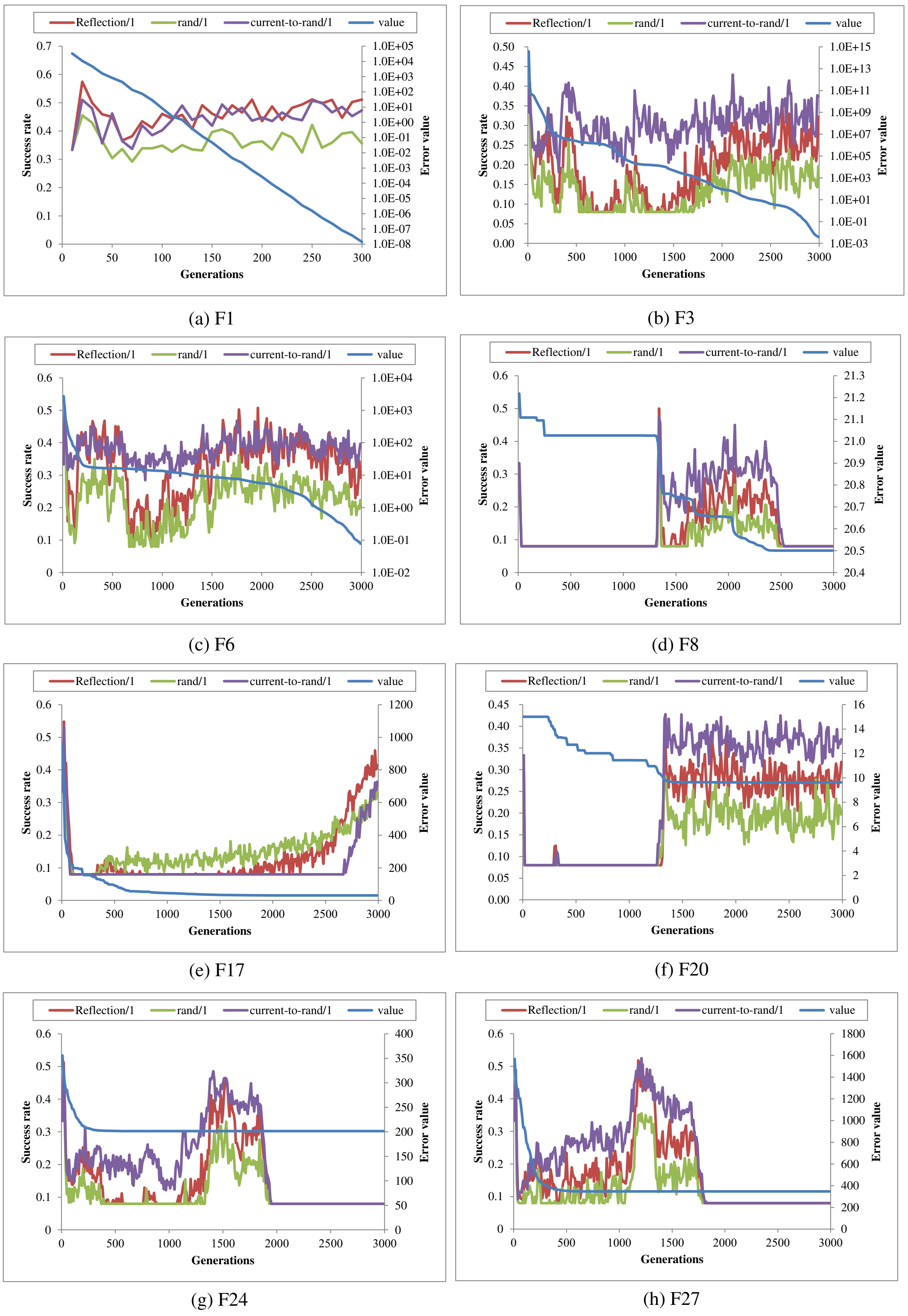


(g) F24



(h) F25

Fig. 3 Comparison of the convergence speeds of HSDE, JADE, SHADE, and MMRDE



(a) F1 (b) F3

(c) F6 (d) F8

(e) F17 (f) F20

(g) F24 (h) F27

Fig. 4 Success rate evolution of three mutation strategies

algorithms can converge globally for functions f_1 and f_5. MMRDE obtains significantly better results than its competitors for f_4. Based on the performance in functions $f_1 \sim f_5$, we can conclude that MMRDE has a strong exploitation ability and achieves the performance level of SHADE in unimodal functions.

2. Basic Multimodal Functions $f_6 \sim f_{20}$: MMRDE performs better than its peers. MMRDE clearly is the best among all the algorithms for functions f_7, f_8, f_{12}, f_{16}, f_{18}, and f_{20}. This shows that the method of using a roulette wheel to choose a mutation strategy seems effective. Our proposed mutation strategy works well with rand/1 and current-to-rand/1. Overall, we see that MMRDE has a strong ability to converge globally.
3. Composition Functions $f_{21} \sim f_{28}$: These hybrid composition functions are made up of several different types of functions and are difficult to optimize. Tables 5, 6, and 7 show that MMRDE achieves similar performance to SHADE and performs better than JADE and HSDE. Specifically, MMRDE obtains the best solution on functions 23, 25, and 26. This shows that MMRDE has a strong ability to solve composition functions.
4. Table 8 shows the results of the application of Wilcoxon's test to all functions. MMRDE is significantly better than JADE and HSDE, but it is not significantly better or worse than SHADE. Both the average ranking and summation of relative errors of MMRDE are the lowest, however, indicating that MMRDE performs better overall than its competitors.

To intuitively reflect the convergence speed of MMRDE relative to other algorithms, we choose a set of results from 30 independent runs to draw the convergence-process curve in eight representative test functions from the benchmark set. These curves are shown in Fig. 3. We see that SHADE has the fastest optimization speed overall. MMRDE obviously converges faster than HSDE. Figure 3c, e and f show that MMRDE seems to have a proper convergence speed to strike a good balance between exploration and exploitation.

5.8 Evolution of mutation strategies

It is interesting to find out which mutation strategies have a higher success rate of generating excellent offspring. A mutation strategy with higher success rates can be allocated to more individuals using the roulette wheel selection method. We select eight representative functions from the benchmark set to represent the success rates of each mutation strategy in different DE stages of one problem. Figure 4 shows the success rate of each mutation strategy and the changes in error value during the evolution.

From Fig. 4, we see that the reflection-based mutation strategy and current-to-rand/1 have higher success rates than mutation strategy rand/1, and they usually dominate the evolutionary process. Mutation strategy rand/1 has less impact on evolution. All strategies have a high success rate in early evolution; however, as evolution proceeds, the success rates of the three mutation strategies vary in different functions. When solving simple problems (such as F1), the success rate of each mutation strategy is high and there is little change in the whole evolution process. But, when a complex function is solved, the success rate of each mutation strategy usually fluctuates greatly. It is worth noting that we set a lower limit on the success rate so that each mutation strategy will not become extinct in some evolutionary stages that were unfavorable to them, and they have the opportunity of dominant evolution at any stage of evolution. Figure 4d, e, and f clearly show the effect of the lower limit of success rate on evolution. The use of roulette options for the three mutation strategies seems to mitigate the process of evolutionary stagnation.

In general, the three strategies used in MMRDE work well, and the algorithm achieves a better balance between exploration and exploitation.

6 Conclusions

In this paper we propose a new mutation operation, called reflection-based mutation, inspired by the reflection operations in the Nelder-Mead method. For each individual to be mutated, we randomly select four individuals from the current population. We then calculate the weighted center of three individuals with better fitness based on their object function values, use the difference vector of the best individual and the worst individual as the perturbation vector, and use the center we have calculated as the target individual, thus completing a mutation operation. We compare DE/Reflection/1 and other mutation strategies in an experiment and prove that the proposed mutation strategy has good exploration and exploitation capability.

Although reflection-based mutation can better balance exploration and exploitation than other compared mutation strategies, it is still a greedy mutation strategy that is prone to premature convergence when solving complex multimodal optimization problems. We add two basic strategies for a population to generate perturbation vectors to maintain population diversity and increase robustness. We use a roulette wheel selection strategy to arrange mutation strategies based on their success rate for each individual. This DE variant, with a combination of multiple mutation strategies based on roulette wheel selection, is named MMRDE. We use a benchmark set, including 28 functions for real-parameter optimization recommended by the Institute of Electrical and Electronics Engineers CEC2013 special session, to test the performance of MMRDE against some improved DE variants. Experimental results indicate that the proposed algorithm can balance between exploration and exploitation. The algorithm shows the effectiveness of three mutation strategies in cooperative work. The proposed algorithm shows it can guide the search for a global optimal solution with fast convergence compared with other improved DE variants.

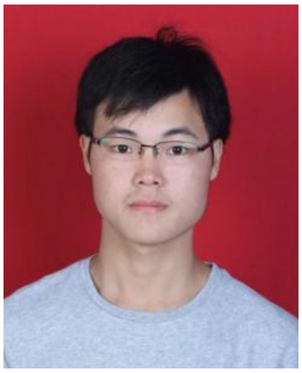
Acknowledgments We thank Miguel Leon (from the School of Innovation, Design and Engineering, Malardalen University) for his great help with this study. We greatly appreciate the reviewers for their thoughtful and encouraging comments on our manuscript, which we all think is helpful for improving the quality of our paper. This study was supported by program 2013KCT-15 of the Shanxi Provincial Key Innovative Research Team and the National Natural Science Foundation of China (51409206). We thank LetPub (www.letpub.com) for its linguistic assistance during the preparation of this manuscript.

References

1. Storn R, Price K (1997) Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim 11(4):341–359
2. Chang CF et al (2007) Robust searching hybrid differential evolution method for optimal reactive power planning in large-scale distribution systems. Electr Power Syst Res 77(5-6):430–437
3. Dragoi EN et al (2013) Optimization methodology based on neural networks and self-adaptive differential evolution algorithm applied to an aerobic fermentation process. Appl Soft Comput 13(1):222–238
4. Iorio AW, Li X (2004) Solving rotated multi-objective optimization problems using differential evolution. AI 2004: Advances in Artificial Intelligence 3339:861–872
5. Das S, Konar A, Chakraborty UK (2005) Two improved differential evolution schemes for faster global search. In: Genetic And Evolutionary Computation Conference, pp 991–998
6. Chakraborty UK, Das S, Konar A (2006) Differential Evolution with Local Neighborhood. In: 2006 IEEE International Conference on Evolutionary Computation 2042–2049
7. Qin AK, Suganthan PN (2005) Self-adaptive differential evolution algorithm for numerical optimization. In: 2005 IEEE Congress on Evolutionary Computation 2:1785–1791
8. Brest J et al (2006) Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. IEEE Trans Evol Comput 10(5):646–657
9. Zhang J, Sanderson AC (2009) JADE: adaptive differential evolution with optional external archive. IEEE Trans Evol Comput 13(5):945–958
10. Mallipeddi R et al (2011) Differential evolution algorithm with ensemble of parameters and mutation strategies. Appl Soft Comput 11(2):1679–1696
11. Wang Y, Cai Z, Zhang Q (2011) Differential evolution with composite trial vector generation strategies and control parameters. IEEE Trans Evol Comput 15(1):55–66
12. Tanabe R, Fukunaga A (2013) Success-history based parameter adaptation for differential evolution. In: 2013 IEEE Congress on Evolutionary Computation 71–78
13. Yi W et al (2015) A new differential evolution algorithm with a hybrid mutation operator and self-adapting control parameters for global optimization problems. Appl Intell 42(4):642–660
14. Leon M, Xiong N (2016) Adapting differential evolution algorithms for continuous optimization via greedy adjustment of control parameters. J Artificial Intell Soft Comput Res 6(2):103–118
15. Wu G et al (2016) Differential evolution with multi-population based ensemble of mutation strategies. Inf Sci 329(C):329–345
16. Leon M, Xiong N (2017) Alopex-based mutation strategy in differential evolution. In: 2017 IEEE Congress on Evolutionary Computation 1978–1984
17. Ronkkonen J, Kukkonen S, Price KV (2005) Real-parameter optimization with differential evolution. In: 2005 IEEE Congress on Evolutionary Computation, vol 1, pp 506–513
18. Price K, Storn RM, Lampinen JA (2005) Differential evolution: a practical approach to global optimization (natural computing series). Springer-Verlag, New York, pp 1–24
19. Jia D, Zheng G, Khurram Khan M (2011) An effective memetic differential evolution algorithm based on chaotic local search. Inf Sci 181(15):3175–3187
20. Ali MZ et al (2017) An Adaptive Multipopulation Differential Evolution with Dynamic Population Reduction. IEEE Trans Cybern 47(7):2768–2779
21. Lampinen J (2002) A Fuzzy Adaptive Differential Evolution Algorithm. Soft Comput 9(5):448–462
22. Ghosh A et al (2011) An improved differential evolution algorithm with fitness-based adaptation of the control parameters. Inf Sci 181(18):3749–3765
23. Mezura-Montes E, Velazquez-Reyes J, Coello Coello CA (2006) A comparative study of differential evolution variants for global optimization. In: 8th Annual Genetic and Evolutionary Computation Conference 485–492
24. Dorronsoro B, Bouvry P (2011) Improving classical and decentralized differential evolution with new mutation operator and population topologies. IEEE Trans Evol Comput 15(1):67–98
25. Islam SM et al (2012) An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization. IEEE Trans Syst Man Cybern B Cybern 42(2):482–500
26. Mohamed AW (2015) An improved differential evolution algorithm with triangular mutation for global numerical optimization. Comput Ind Eng 85(C):359–375
27. Wang S et al (2017) Self-adaptive differential evolution algorithm with improved mutation strategy. Soft Computing, pp 1–15. Springer Berlin Heidelberg. https://doi.org/10.1007/s00500-017-2588-5. Online ISSN: 1433-7479
28. Piotrowski AP (2013) Adaptive memetic differential evolution with global and local neighborhood-based mutation operators. Inf Sci 241(10):164–194
29. Gong W et al (2011) Adaptive strategy selection in differential evolution for numerical optimization: An empirical study. Inf Sci 181(24):5364–5386
30. Tasoulis DK, Plagianakos VP, Vrahatis MN (2005) Clustering in Evolutionary Algorithms to efficiently compute simultaneously local and global minima. In: 2005 IEEE Congress on Evolutionary Computation, vol 2, pp 1847–1854
31. Epitropakis MG, Plagianakos VP, Vrahatis MN (2008) Balancing the exploration and exploitation capabilities of the differential evolution algorithm. In: 2008 IEEE Congress on Evolutionary Computation 2686–2693
32. Ali MZ, Awad NH, Suganthan PN (2015) Multi-population differential evolution with balanced ensemble of mutation

strategies for large-scale global optimization. Appl Soft Comput 33(C):304–327

33. Piotrowski AP, Napiorkowski JJ, Kiczko A (2012) Differential Evolution algorithm with Separated Groups for multi-dimensional optimization problems. Eur J Oper Res 216(1):33–46
34. Qin AK, Huang VL, Suganthan PN (2009) Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Trans Evol Comput 13(2):398–417
35. Das S et al (2009) Differential evolution using a neighborhood-based mutation operator. IEEE Trans Evol Comput 13(3):526–553
36. Pencheva T, Atanassov K, Shannon A (2009) Modelling of a roulette wheel selection operator in genetic algorithms using generalized nets. Int J Bio 13(4):101–105
37. Lipowski A, Lipowska D (2012) Roulette-wheel selection via stochastic acceptance. Physica A: Stat Mech Appl 391(5):2193–2196
38. Ho-Huu V et al (2018) An improved differential evolution based on roulette wheel selection for shape and size optimization of truss structures with frequency constraints. Neural Comput Applic 29(1):167–185
39. Peng F et al (2009) Multi-start JADE with knowledge Transfer for numerical optimization. In: 2009 IEEE Congress on Evolutionary Computation 1889–1895
40. Liang J et al (2013) Problem definitions and evaluation criteria for the CEC 2013 Special Session on Real-Parameter Optimization
41. Alcala-Fdez J et al (2009) KEEL: A software tool to assess evolutionary algorithms for data mining problems. Soft Comput 13(3):307–318



Dr. Wuwen Qian was born in 1991 and is presently working towards his PH.D. Degree at the Xi'an University of Technology. His main research interests are Optimization algorithms, Percolation mechanics of rock and soil, and Estimation of percolation parameters.

Junrui Chai was born in 1968 and is a professor with institute of water resources and hydro-electric engineering, Xi'an University of Technology, Shaanxi, China. Ph.D. Degree from Xi'an University of Technology, Shaanxi, China, in 2000. His main research interests are Numerical analysis of hydro-structure, Hydraulic seepage mechanics, and Geotechnical fluid mechanics.

Zengguang Xu was born in 1982 and is a professor with institute of water resources and hydro-electric engineering, Xi'an University of Technology, Shaanxi, China. Ph.D. Degree from Shanghai Jiao Tong University, Shanghai, China, in 2012. His main research interests are Seepage and groundwater simulation of water conservancy project, Percolation characteristics of special rock mass and soil, and Estimation of percolation parameters.

Ziying Zhang was born in 1993 and is presently working towards her Master's degree at the Xi'an University of Technology. Her main research interests are Differential evolution algorithm and Slope stability analysis.