Soft Comput (2014) 18:2209–2225
DOI 10.1007/s00500-013-1195-3

METHODOLOGIES AND APPLICATION

Enhancing differential evolution with role assignment scheme

Xinyu Zhou · Zhijian Wu · Hui Wang · Shahryar Rahnamayan

Published online: 6 December 2013

Abstract As one of the most popular evolutionary algorithms, differential evolution (DE) has been used for solving a wide range of real-world problems. The performance of DE highly depends on the chosen mutation strategy and control parameter settings. Although the conventional trial-and-error procedure can be used to elaborately select the proper strategy and to tune the parameter values, this procedure is often very time-consuming and is not suitable for practitioners without a priori experience. To tackle this problem, DE with a novel role assignment (RA) scheme is proposed in this paper. In the RA scheme, both the fitness information and positional information of individuals are utilized to dynamically divide the population into several groups. Each group is considered as a role, which has its own mutation strategy and parameter settings and is expected to play a different role in the evolution process. To verify the performance of our approach, experiments are conducted on 23 well-known benchmark functions. Results show that our approach is better than, or at least comparable to, several state-of-the-art DE variants.

Keywords Differential evolution · Mutation strategy · Control parameter settings · Role assignment scheme

Communicated by Z. Zhu.

X. Zhou (✉) · Z. Wu
State Key Lab of Software Engineering, School of Computer, Wuhan University, Wuhan 430072, China
e-mail: xyzhou@whu.edu.cn

Z. Wu
e-mail: zhijianwu@whu.edu.cn

H. Wang
School of Information Engineering, Nanchang Institute of Technology, Nanchang 330099, China
e-mail: huiwang@whu.edu.cn

S. Rahnamayan
Department of Electrical, Computer and Software Engineering, University of Ontario Institute of Technology (OUIT), 2000 Simcoe Street North, Oshawa, ON L1H 7K4, Canada
e-mail: shahryar.rahnamayan@uoit.ca

1 Introduction

Differential evolution (DE), proposed by Storn and Price in 1995 (Storn and Price 1997; Price et al. 2005), is one of the most powerful evolutionary algorithms (EAs), which has been successfully applied to a wide variety of practical cases, such as engineering optimal design, digital filter design, image processing, and data mining (Maulik and Saha 2010; Das et al. 2008). Similar to other EAs, DE employs three evolutionary operations: mutation, crossover and selection, to evolve its population toward the global optimum. For mutation and crossover operations, the performance of DE is sensitive to mutation strategy and the associated control parameters, such as the scaling factor F and crossover rate CR. In fact, different kinds of optimization problems generally require different mutation strategies and parameter values to maximize the performance of DE. Although the conventional trial-and-error procedure can be used to elaborately select a proper mutation strategy and to tune parameter values, this procedure is often very time-consuming and is not suitable for practitioners without a priori experience. Hence, how to dynamically select a mutation strategy and set the control parameters for DE has become a challenging research topic.

In the past decade, a number of approaches for dynamically selecting mutation strategies and/or setting the control parameters have been proposed to enhance the robustness of DE (Liu and Lampinen 2005; Brest et al. 2006; Qin et al.

2009; Mallipeddi et al. 2011; Ghosh et al. 2011; Wang et al. 2011b). Most of these approaches can be considered as adaptive or self-adaptive schemes (Eiben et al. 1999; Das and Suganthan 2010; Neri and Tirronen 2010), which have shown faster and more reliable convergence performance than the classic DE. A common characteristic among these methods is that the previous experiences of generating promising candidate solutions in term of fitness values, as the feedback from the evolutionary search, are utilized to select the most suitable mutation strategy and/or parameter settings to guide the search. Inspired by these techniques, the fitness information of the population is utilized to assign mutation strategies with distinct parameter settings to different individuals in our approach. Moreover, during the evolution in the DE, different individuals of the population usually move through different regions of search space on the landscape. For these individuals, the performance of their associated mutation strategies and parameter settings is influenced by the characteristics of the regions being explored by them (Gong et al. 2011b). Therefore, the positional information of individuals in the search landscape is utilized in our approach as well as the fitness information. By simultaneously considering the fitness information and positional information of the population, we propose a role assignment (RA) scheme for DE, in which each individual of the population is assigned with a role. Each role has its own mutation strategy and parameter settings, which aims to play a different role in the evolution by exhibiting distinct search behavior. The RA scheme based DE algorithm is called RADE, which has a simple structure and thus is easy to implement. Comprehensive experiments have been conducted to verify its effectiveness and efficiency, and also it is compared with four other DE schemes and five other state-of-the-art DE variants.

The rest of this paper is organized as follows. In Sect. 2, the classic DE is briefly introduced. Section 3 reviews the related works on DE. The proposed RA scheme and RADE algorithm are described in Sect. 4 in detail. Experimental results are reported in Sect. 5. Finally, this study is concluded in Sect. 6.

2 Classic differential evolution

DE is a population-based stochastic search algorithm for global numerical optimization. Similar to other EAs, DE starts with a population of *NP* vectors representing the candidate solutions, where *NP* indicates the population size. Let us assume that $X_i^G = [x_{i,1}^G, x_{i,2}^G, \ldots, x_{i,D}^G]$ is the ith candidate solution vector at the Gth generation, where $i = 1, 2, \ldots, NP$, and D is the problem dimension. The framework of DE mainly consists of the following three evolutionary operations: mutation, crossover, and selection.

2.1 Mutation operation

At each generation G, DE creates a mutant vector (or donor vector) $V_i^G = [v_{i,1}^G, v_{i,2}^G, \ldots, v_{i,D}^G]$ for each individual (or target vector) $X_i^G = [x_{i,1}^G, x_{i,2}^G, \ldots, x_{i,D}^G]$ in the current population, and the following are five well-known mutation strategies.

- DE/rand/1

$$V_i^G = X_{r1}^G + F \cdot (X_{r2}^G - X_{r3}^G) \tag{1}$$

- DE/rand/2

$$V_i^G = X_{r1}^G + F \cdot (X_{r2}^G - X_{r3}^G) + F \cdot (X_{r4}^G - X_{r5}^G) \tag{2}$$

- DE/best/1

$$V_i^G = X_{best}^G + F \cdot (X_{r1}^G - X_{r2}^G) \tag{3}$$

- DE/best/2

$$V_i^G = X_{best}^G + F \cdot (X_{r1}^G - X_{r2}^G) + F \cdot (X_{r3}^G - X_{r4}^G) \tag{4}$$

- DE/current-to-best/1

$$V_i^G = X_i^G + F \cdot (X_{best}^G - X_i^G) + F \cdot (X_{r1}^G - X_{r2}^G) \tag{5}$$

In the above equations, the indices $r1, r2, r3, r4$ and $r5$ are mutually exclusive integers which are randomly chosen from $[1, NP]$, and they all are different from the base index i. The scaling factor F is a real number that controls the mutation step of the difference vector. X_{best}^G is the best individual in terms of fitness value at the current generation G. In addition to the above listed mutation strategies, there exists several other ones in some improved DE variants, such as the "DE/current-to-*p*best" (Zhang and Sanderson 2009), the proximity-based mutation operators (Epitropakis et al. 2011), and the ranking-based mutation operators (Gong and Cai 2013).

2.2 Crossover operation

The crossover operation aims to enhance diversity of the population, which takes place after a mutant vector is generated through mutation operator (Das and Suganthan 2010). In this operation, a trial vector $U_i^G = [u_{i,1}^G, u_{i,2}^G, \ldots, u_{i,D}^G]$ is constructed by mixing the components of a mutant vector V_i^G with that of its target vector X_i^G. The widely used binomial crossover is formulated as follows.

$$u_{i,j}^G = \begin{cases} v_{i,j}^G & \text{if } rand_j(0,1) \le CR \text{ or } j = j_{rand} \\ x_{i,j}^G & \text{otherwise} \end{cases} \tag{6}$$

where $rand_j(0, 1)$ is a anew generated random number uniformly sampled from [0, 1] for each dimension, and $CR \in [0, 1]$ is called the crossover rate. The j_{rand} is a randomly chosen integer from $[1, D]$, which can guarantee that the trial vector differs from its target vector. Another commonly used crossover operation is exponential crossover (Neri and Tirronen 2010; Das and Suganthan 2010; Wang et al. 2012).

2.3 Selection operation

After the crossover operation, a greedy selection mechanism is used to select the better one from the target vector X_i^G and the trial vector U_i^G according to their fitness values $f(\cdot)$. If, and only if, the trial vector is better than the target vector, X_i^{G+1} is set to U_i^G; otherwise, X_i^{G+1} is kept the same with X_i^G.

$$X_i^{G+1} = \begin{cases} U_i^G & \text{if } f(U_i^G) < f(X_i^G) \\ X_i^G & \text{otherwise.} \end{cases} \tag{7}$$

3 Related works

Although the classic DE has shown powerful performance in solving benchmark and real-world problems, its performance highly depends on the chosen mutation strategy and control parameter settings. Therefore, considerable efforts have been devoted to improve the robustness and convergence performance of DE by dynamically selecting the mutation strategy and/or setting the control parameters, during the last decade. In this section, a brief overview of the relative works is presented as follows.

Liu and Lampinen (2005) proposed a fuzzy adaptive DE (FADE), which uses a fuzzy logic control approach to adapt the scaling factor F and crossover rate *CR*. The results on high dimensional problems show that FADE outperforms the classic DE, and shows a faster convergence rate. Brest et al. (2006) presented a self-adaptive DE (jDE), in which the control parameters F and *CR* are encoded into the individual and then are adjusted by introducing two new parameters τ_1 and τ_2. The better values of these encoded control parameters lead to better individuals which, in turn, are more likely to survive and produce offspring and, hence, propagate these better parameter values. In order to eliminate the need of manual tuning of control parameters, Salman et al. (2007) investigated a self-adaptive DE (SDE), in which the scaling factor F is adapted by using the mechanism proposed in Abbass (2002), where F is calculated as a stochastic linear combination of the scaling factors of randomly selected individuals. The values of *CR* are generated by sampling from a normal distribution $N(0.5, 0.15)$. Moreover, the performance of SDE using the ring neighborhood topology is also investigated in Salman et al. (2007). Zhang and Sanderson (2009) proposed an adaptive DE with optional external archive (JADE). In JADE, the scaling factor F and crossover rate *CR* are generated according to a normal distribution $N(\mu_{CR}, 0.1)$ and a Cauchy distribution $C(\mu_F, 0.1)$, respectively. Both of the parameters μ_{CR} and μ_F are related to the successful experiences of generating good offspring entering the next generation, and they all are initialized to 0.5 at first. Ghosh et al. (2010, 2011) proposed a simple yet effective adaptation technique, fitness-based adaptation scheme, for tuning F and *CR*, and the resulting algorithm was called FiADE. In their approach, each individual in the population has its own settings for F and *CR*, which can be considered as a dither method according to the classification defined by Price et al. (2005). The fitness-based adaptation schemes are based on the objective function values or fitness values of the target vectors and donor vectors. The presented results show that the performance of FiADE is very competitive compared with other DE variants.

In addition to dynamical setting of the control parameters, the following approaches also focus on dynamical selection of mutation strategies. Qin and Suganthan (2005) and Qin et al. (2009) proposed a self-adaptive DE (SaDE), in which multiple mutation strategies are adaptively adjusted through learning from the previous successful experiences. The control parameters F and *CR* are also adaptively set by sampling their values from normal distributions with different mean parameters. Mallipeddi et al. (2011) proposed to employ an ensemble of mutation strategies and control parameters with DE (EPSDE). In EPSDE, multiple mutation strategies with distinct characteristics are adopted to form a strategy candidate pool, and similarly, two parameter candidate pools are constructed for F and *CR*. Then, the combinations of different mutation strategies and parameter values which produced better offspring are stored to guide the search process. The experimental results showed that EPSDE outperforms SaDE, jDE, and JADE on a majority of test problems. Gong et al. (2011b) proposed a strategy adaptation mechanism (SaM) to choose a more suitable strategy from a strategy candidate pool constructed by four different strategies. The SaM is integrated into the framework of JADE, and the resulting algorithm is called SaJADE. The experiments conducted on 20 benchmark problems and two real-world problems show that SaJADE is promising. Wang et al. (2011b) proposed a composite DE (CoDE) which utilizes a strategy candidate pool and a parameter candidate pool. CoDE generates three trial vectors at the same time by using three strategies from the strategy candidate pool with randomly selected parameter values from the parameter candidate pool, then the best one among the three trial vectors will head to the selection operation. Very recently, Wang et al. (2013) proposed a Gaussian bare-bones DE (GBDE) in order to minimize the effects of DE control parameters. In the GBDE, a Gaussian mutation strategy is defined by sampling the search space between the best individual and the current one. Furthermore, the classic mutation strategy, DE/best/1, is combined with the Gaussian mutation strategy to improve the convergence speed of GBDE, and this modified version is called MGBDE. The reported empirical results verified that MGBDE achieves better performance than GBDE and some other state-of-the-art DE variants on many unimodal and multimodal functions.

Furthermore, some other operations are incorporated into DE. Sun et al. (2005) developed DE/EDA, which combines DE with estimation of distribution algorithm (EDA). Noman and Iba (2008) incorporated an adaptive local search (LS) operation into the classic DE. Rahnamayan et al. (2008) proposed an opposition-based DE (ODE) which employs opposition-based learning (OBL) strategy for population initialization and for generating new solutions. The experimental results indicate that the convergence speed and the solution accuracy of DE can be clearly improved by making use of the OBL strategy. Similarly, Wang et al. (2011a) proposed an improved version of OBL, i.e. generalized opposition-based learning (GOBL). The GOBL strategy is incorporated into DE to solve high-dimensional problems efficiently. Caponio et al. (2009) proposed a super-fit memetic DE (SFMDE), in which PSO and two other local searchers are incorporated. DE/BBO was proposed by Gong et al. (2011a), which combines DE with biogeography-based optimization (BBO) algorithm.

In this section, we only presented a brief overview of some related works, interested readers can refer to two comprehensive surveys of DE in Das and Suganthan (2010) and Neri and Tirronen (2010).

4 The proposed approach

As we all know, the performance of DE highly depends on the chosen mutation strategy and parameter settings. For different problems at hand, DE realizations employing different mutation strategies and parameter settings usually perform differently. Therefore, over the past decade, there is an increasing research trend in the DE community that, adaptive or self-adaptive techniques are developed to select a proper mutation strategy and to set suitable parameter values for DE during its evolutionary search. The reported results show the superior performance of these techniques, which greatly improve the robustness of DE. It is worth to note that a common characteristic of these adaptive or self-adaptive techniques is that, the previous experiences of generating promising candidate solutions in term of fitness values, as the feedback from the evolutionary search, are utilized to guide the search process. Inspired by this, the fitness information of the population is utilized in our approach to allocate mutation strategies with variuos parameter settings for different individuals. Moreover, during the evolution of DE, different individuals of the population usually move through different regions in the search landscape. For these individuals, the performance of their associated mutation strategies and parameter settings is influenced by the characteristics of the regions being explored by them (Gong et al. 2011b). Therefore, this motivates us to utilize the the positional information of individuals as well as the fitness information in our approach.

4.1 The role assignment scheme

In the RA scheme, the population is divided into three groups by considering both the fitness information and positional information of its individuals. The three groups are assigned with three different roles: *exploiter*, *follower*, and *explorer*. Each role is desired to exhibit different search behavior by employing different mutation strategies and parameter settings. The three roles are explained as follows.

Exploiters are expected to be the individuals which have good fitness values, and are far away from the best individual of current population in the search landscape. It is possible that they locate on/around different local optimal regions when solving a multimodal function. The objective of this role is to exhibit local search for finding better candidate solutions in its neighborhood. Followers are expected to be the individuals which also have good fitness values but close to the best individual of current population in the search landscape. It is possible that they locate on/around the same local optimal region with the best individual when solving a multimodal function. In order to take the advantages of their good fitness values, the search behavior of the followers is also focused on their neighborhood. The remaining individuals in the population are identified as explorers. Obviously, the explorers may neither have good fitness values nor near the best individual of current population in the search landscape. It can be inferred from the name that explorers aim to explore new regions for diversifying the population.

The following steps describe how to divide the population into three groups.

Step 1. Calculate the fitness difference between each individual X_i and the best individual X_b, and indicate it as Δf_i ($\Delta f_i = f(X_i) - f(X_b)$). Then, all the Δf_i can be collected to calculate the mean value Δf_{mean} as follows.

$$\Delta f_{mean} = \frac{1}{NP-1}\sum_{i=1}^{NP}(f(X_i) - f(X_b)) \tag{8}$$

where *NP* is the population size, $f(x_i)$ denotes the fitness value of X_i, and the fitness value of X_b is denoted as $f(x_b)$. Note that in the above equation when i equals to the index of X_b, the Δf_i is zero. After getting the Δf_{mean}, a set **A** is defined to contain those individuals of which the Δf_i are all smaller than Δf_{mean}. Mathematically, the set **A** can be formulated according to the next equation.

$$\mathbf{A} = \{X_i | \Delta f_i \le \Delta f_{mean},\ i = 1, 2, \ldots, NP\} \tag{9}$$

Step 2. Similar to the Δf_i and Δf_{mean}, Δd_i is used to indicate the Euclidean distance from each individual X_i to the best individual X_b, and Δd_{mean} represents the mean Euclidean distance.

$$\Delta d_{mean} = \frac{1}{NP-1}\sum_{i=1}^{NP}\sqrt{\sum\nolimits_{j=1}^{D}(x_{i,j}-x_{b,j})^2} \qquad (10)$$

where $x_{i,j}$ and $x_{b,j}$ are the jth variable values of X_i and X_b, respectively. Then, we select the individuals having smaller Δd_i than Δd_{mean} to construct a new set **B**. Similar to the set **A**, **B** can also be formally defined as follows.

$$\mathbf{B} = \{X_i | \Delta d_i \le \Delta d_{mean},\ i = 1, 2, \ldots, NP\} \qquad (11)$$

Step 3. It can be inferred that all the individuals contained in the set **A** have better fitness values than the remaining individuals in the current population **P**. According to the explains of exploiter and follower, the positions of those individuals in the set **A** may be one of the two cases: (1) they cluster around the best individual; or (2) they are far away from the best individual and even on/around different local optimal regions when the problem is multimodal. But in the set **B**, all the individuals are closer to the best individual in the search landscape than the remaining ones in the population. It means that the individuals in the set **B** may include the ones which have good fitness values and also close to the best individual. Recognizing the features of set **A** and **B**, the exploiters and followers can be picked out from the population by implementing set operations on **A** and **B**. The following Eqs. (12) and (13) show the formal definitions for the exploiters and followers, respectively. A set **C** is defined to represent the exploiters, and a set **D** is defined to denote the followers.

$$\mathbf{C} = \mathbf{A} - \mathbf{B} \qquad (12)$$

$$\mathbf{D} = \mathbf{A} \cap \mathbf{B} \qquad (13)$$

As seen from the above two Eqs. (12) and (13), the set **C** contains the individuals which have good fitness values but are far away from the best individual. While the set **D** contains the individuals that have good fitness values and also close to the best individual. Moreover, it is worth to note that the best individual of current population is always included in the set **C**. For the explorers, the following equation is used to define them.

$$\mathbf{E} = \mathbf{P} - (\mathbf{C} \cup \mathbf{D}) \qquad (14)$$

where the set **E** is defined to represent the explorers, and the set **P** is the current population. It can be seen that the individuals that are not contained in both **C** and **D** are identified as the explorers. It is obviously that these individuals may neither have good fitness values nor close to the best individual in the search landscape.

The relationships among the aforementioned sets are described in Venn diagram in Fig. 1, where the rectangle denotes the whole population **P**, and the two ellipses represent the set **A** and **B**. The areas marked with horizontal lines and vertical lines in the set **A** denote the set **C** and **D**, respectively. The blank areas in the set **P** (including parts of the set **B**) indicate the set **E**.

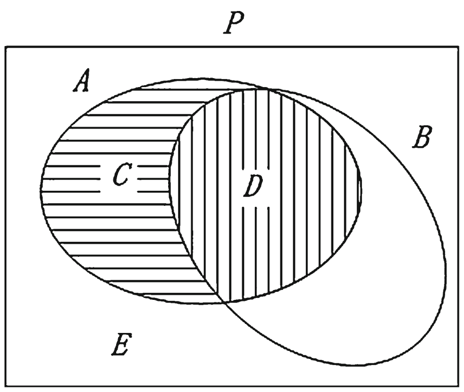

Fig. 1 The relationships among different sets

4.2 Selection of mutation strategies and parameter settings

After defining the three roles, it is important to select proper mutation strategies and parameter settings for them. According to their objectives, the following combinations of mutation strategies and parameter settings are selected for the three roles.

- Exploiter
  DE/best/1 (F = 0.5 and CR = 0.1)
- Follower
  DE/rand/1 (F = 0.6 and CR = 0.1)
- Explorer
  DE/rand/1 (F = 0.7 and CR = 0.9)

The explanations for such selection can be listed as follows.

(a) For the exploiters, their search behavior is focused on their neighborhood, because they have good fitness values, which implies that better candidate solutions may locate around their positions in the search landscape. Therefore, the DE/best/1 strategy is selected for them, since this strategy is helpful to exploit the neighborhood and can accelerate convergence (Yang et al. 2008). For the associated control parameters, it is known that a large value of F can make the mutant vectors distribute widely in the search space and can increase the population diversity. In contrast, a small value of F makes the search focus on the neighborhood, and thus it can speed up the convergence (Wang et al. 2011b). Similar to F, a large CR value can also be helpful to encourage the diversity, but a small one can facilitate the convergence (Zaharie 2009).

So, in order to satisfy the objectives of the exploiters, *F* is set to 0.5 and *CR* is set to 0.1. These two values are also suggested by other researchers, such as Storn and Price (1997) suggested that *F* should be 0.5 as an good initial choice and *CR* can be set to 0.1 or 0.9, and Ronkkonen et al. (2005) suggested that *F* should be chosen from the range [0.4, 0.95], and *CR* should be between [0.0, 0.2] or [0.9, 1.0].

(b) For the followers, although their objectives are also to search their neighborhood, the DE/rand/1 strategy, more explorative than the DE/best/1 strategy, is selected for them. According to the definition, the followers may locate around the best individual in the search landscape. If the DE/best/1 strategy is used, the available search region of the followers may overlap with that of the best individual. Therefore, in order to expand the search regions, the DE/rand/1 strategy is used and the value of *F* is also slightly larger than the exploiters. The value of *CR* is still the same as that of the exploiters, which is helpful to focus the search on the neighborhood.

(c) For the explorers, the DE/rand/1 strategy is also selected, because this strategy is the most commonly used one which has no bias to any special search directions, and hence is very robust. But compared with the followers, both of the *F* and *CR* values are larger, because the objectives of the explorers are to explore new regions and to maintain the population diversity for powerful ability in global exploration.

4.3 The framework of RADE

The framework of RADE is described in the following Algorithm 1, where *FEs* represents the number of fitness evaluations, and the *MaxFEs* is the maximum number of fitness evaluations. *NP* is the population size.

Note that the number of individuals of different roles changes dynamically during the evolution process. For better clarification, the 2-D Shekel's Foxholes function illustrated in Fig. 2 is employed as a case study. This function has 24 distinct local minima and one global minimum $f(-32, -32) = 0.998004$, in the range $[65.536, 65.536]^2$, its detailed definition can be found in Yao's literature (Yao et al. 1999). In this case study, the RADE is used to solve the Shekel's Foxholes problem to show its search behavior. For the control parameters, *NP* is set to 30, and the others are the same as description in the Sect. 4.2.

Figure 3 shows the contour plots of the Shekel's Foxholes function and the distribution of different roles at the first, 5th, 10th, and 20th generations. The markers: triangle, circle and star denote the exploiters, followers, and explorers, respectively; besides, the triangle filled with blue color is the best individual. At the beginning of evolution, all the individuals almost cover the entire search space due to the uniform random initialization. At this stage, the exploiters and followers are almost locate at or around different local optimal regions. Then, the search regions become smaller and smaller as the evolution proceeds. It is important to note that different exploiters are usually far away from the best individual while the followers are much closer to the best individual, and the explorers distribute on "poorer" search regions. Moreover, it can be seen that the number of explorers is decreasing with the evolution, while the number of exploiters and followers is increasing. This implies that the search behavior of RADE gradually transforms from exploration to exploitation as the number of generations increases.

Algorithm 1 The framework of RADE

```
1: Generate an initial population $P$;
2: Evaluate the fitness value of each individual $X_i$;
3: $FEs = NP$;
4: while $FEs \le MaxFEs$ do
5:     Select the best individual as $X_b$ from $P$;
6:     Calculate the mean fitness difference $\Delta f_{mean}$ according to Eq. (8);
7:     Calculate the mean Euclidean distance $\Delta d_{mean}$ according to Eq. (10);
8:     According to Eqs. (12)-(14), divide the population $P$ into the exploiters, followers and explorers;
9:     for $i = 1$ to $NP$ do
10:        According to its role, generate a trial vector for $X_i$ by using the corresponding mutation strategy and associated parameter values;
11:        Select the better one from the trial vector and $X_i$ to enter the next generation;
12:    end for
       $FEs = FEs + NP$;
13: end while
```

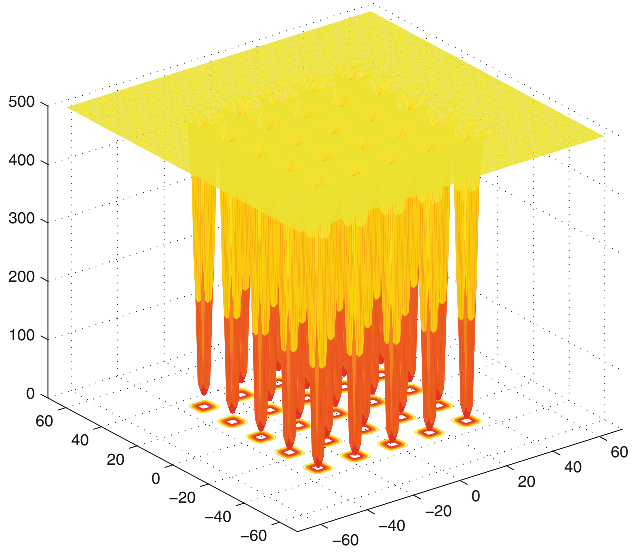

Fig. 2 3-D plot of the Shekel's Foxholes function

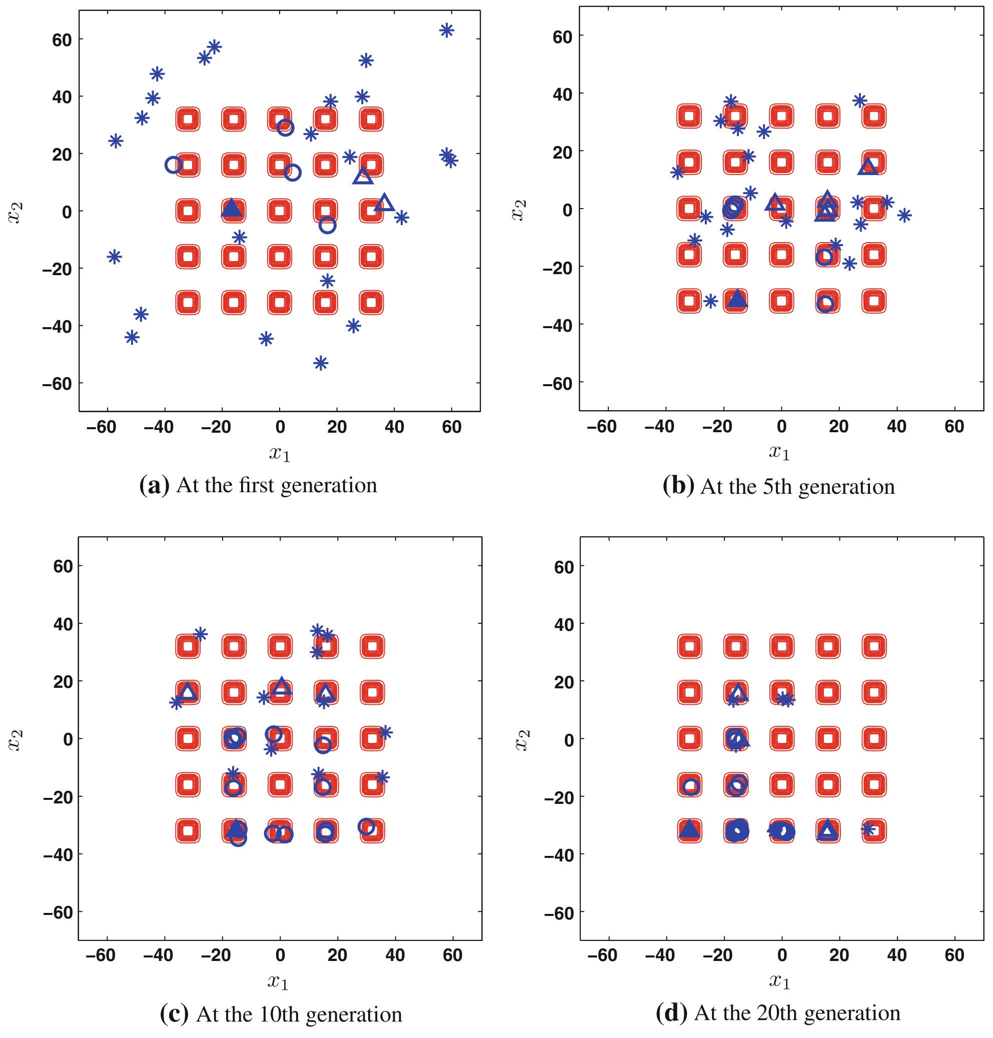

Fig. 3 The distribution of different roles at different generations

5 Experimental verification

5.1 Benchmark functions

In order to verify the performance of the proposed RADE algorithm, a set of 23 benchmark functions is used in the following experiments. The first thirteen are well-known scalable functions (Yao et al. 1999). Among these functions, F01–F04 are unimodal functions. F05 is the Rosenbrock function, which is multimodal when $D > 3$ (Shang and Qiu 2006). F06 is a step function which has one minimum and it is discontinuous, while F07 is a noisy quartic function. F08–F13 are multimodal functions with many local minima. The remaining ten shifted and rotated benchmark functions (F14–F23) are from CEC 2005 competition (Suganthan et al. 2005). Brief descriptions of these functions are summarized in Table 1.

5.2 Comparison of RADE with other DE schemes

In this section, in order to evaluate the effectiveness of RADE, three different DE schemes (DE1–DE3) are compared with RADE, which are selected for the three roles, respectively. Moreover, the most commonly used DE scheme (DE4) is also included as a baseline for comparison.

(a) DE1: best/1/bin ($F = 0.5$ and $CR = 0.1$);
(b) DE2: rand/1/bin ($F = 0.6$ and $CR = 0.1$);
(c) DE3: rand/1/bin ($F = 0.7$ and $CR = 0.9$);
(d) DE4: rand/1/bin ($F = 0.5$ and $CR = 0.9$).

This current experiment includes two aspects: (1) quality of the final solutions; and (2) scalability test. For common parameters, the above four contestant DE schemes use the same settings as RADE, such as the same population size

Table 1 Twenty-three benchmark functions used in our experimental verification

Functions	Name	Search range	Global optimum
F01	Sphere	[−100, 100]	0
F02	Schwefel 2.22	[−10, 10]	0
F03	Schwefel 1.2	[−100, 100]	0
F04	Schwefel 2.21	[−100, 100]	0
F05	Rosenbrock	[−30, 30]	0
F06	Step	[−100, 100]	0
F07	Quartic with noise	[−1.28, 1.28]	0
F08	Schwefel 2.26	[−500, 500]	$-418.98 \cdot D$
F09	Rastrigin	[−5.12, 5.12]	0
F10	Ackley	[−32, 32]	0
F11	Griewank	[−600, 600]	0
F12	Penalized 1	[−50, 50]	0
F13	Penalized 2	[−50, 50]	0
F14	Shifted sphere	[−100, 100]	−450
F15	Shifted Schwefel's problem 1.2	[−100, 100]	−450
F16	Shifted rotated high conditioned elliptic function	[−100, 100]	−450
F17	Shifted Schwefel's problem 1.2 with noise in fitness	[−100, 100]	−450
F18	Schwefel's problem 2.6 with global optimum on bounds	[−100, 100]	−310
F19	Shifted Rosenbrock	[−100, 100]	390
F20	Shifted rotated Griewank's function without bounds	[0, 600]	−180
F21	Shifted rotated Ackley's function with global optimum on bounds	[−32, 32]	−140
F22	Shifted Rastrigin's function	[−5, 5]	−330
F23	Shifted rotated Rastrigin's function	[−5, 5]	−330

and stopping criterion. The detailed parameter settings are listed as follows.

(a) Population size: $NP = 30$.
(b) Stop criterion: There are two cases: (1) for the first experiment, each algorithm stops when the number of fitness evaluations (*FEs*) reaches the maximum value. For the first thirteen functions, the maximum value ($MaxFEs$) is set to 200,000, but for the remaining ten CEC 2005 functions, $MaxFEs$ is equal to 300,000; and (2) for the second experiment, $MaxFEs$ is set to $5{,}000 \cdot D$ for the first 13 functions, but $MaxFEs$ is set to $10{,}000 \cdot D$ for the remaining functions, where D is the dimension of the problem.
(c) Number of runs: each algorithm is run 30 times per function.

5.2.1 Comparison of final solution's quality

The results of all functions for $D = 30$ are shown in Table 2, where "Mean error" indicates the mean function error value, and "Std dev" represents the corresponding standard deviation. In order to compare the significance between two algorithms, the paired Wilcoxon signed-rank test is used. The Wilcoxon signed-rank test is a non-parametric statistical hypothesis test, which can be used as an alternative to the paired t test when the results cannot be assumed to be normally distributed (Garca et al. 2009, 2010). In Table 2, according to the Wilcoxon's test, the results are summarized as "*w/l/t*", which denotes that RADE wins on w functions, loses on l functions, and ties on t functions, compared with its corresponding competitor. "†", "‡" and "≈" indicate our approach is respectively better than, worse than, or similar to its competitor according to the Wilcoxon signed-ranked test at $\alpha = 0.05$.

With respect to the overall performance, from Table 2, we can see that, our approach achieves better results than other algorithms on the majority of test functions. To be specific, RADE outperforms DE1 on 16 out of 23 functions and only loses on three functions. On the unimodal functions F01 and F02, DE1 achieves the best performance, due to its greedy strategy. It is worth to note that, although the mean values of DE1 on F12 and F13 are much worse than RADE, the performance between DE1 and RADE are similar

Table 2 Comparisons of mean function error values and standard deviation for all test functions at $D = 30$

Function	DE1 Mean error ± standard deviation	DE2 Mean error ± standard deviation	DE3 Mean error ± standard deviation	DE4 Mean error ± standard deviation	RADE Mean error ± standard deviation
F01	2.19E−148 ± 4.33E−148‡	4.78E−84 ± 4.43E−84‡	5.30E−45 ± 1.79E−44†	5.30E+00 ± 1.36E+01†	6.64E−76 ± 1.66E−75
F02	1.33E−83 ± 1.06E−83‡	9.74E−49 ± 5.89E−49‡	5.49E−24 ± 9.33E−24†	2.12E−10 ± 1.14E−09≈	5.19E−43 ± 3.37E−43
F03	4.12E+01 ± 2.34E+01†	1.53E+03 ± 3.52E+02†	4.58E−08 ± 9.90E−08‡	5.85E−02 ± 2.55E−01†	1.94E−07 ± 2.45E−07
F04	2.59E+01 ± 5.37E+00†	4.17E−09 ± 1.37E−09‡	4.15E+00 ± 3.47E+00†	1.96E+01 ± 5.22E+00†	2.58E−07 ± 9.42E−08
F05	4.38E+01 ± 2.62E+01†	2.94E+01 ± 1.04E+01†	4.83E+00 ± 1.67E+01†	3.62E+03 ± 7.36E+03†	4.42E−11 ± 1.56E−10
F06	6.67E−02 ± 2.49E−01≈	0.00E+00 ± 0.00E+00≈	0.00E+00 ± 0.00E+00≈	0.00E+00 ± 0.00E+00≈	0.00E+00 ± 0.00E+00
F07	3.30E−03 ± 1.07E−03‡	5.26E−03 ± 9.64E−04≈	6.31E−03 ± 2.02E−03†	1.08E−02 ± 6.93E−03†	4.60E−03 ± 1.18E−03
F08	3.74E+02 ± 2.32E+02†	3.82E−04 ± 0.00E+00‡	1.03E+03 ± 7.47E+02†	1.44E+03 ± 4.72E+02†	3.16E+01 ± 6.06E+01
F09	2.72E+00 ± 1.54E+00†	3.32E−02 ± 1.79E−01≈	2.77E+01 ± 9.40E+00†	1.88E+01 ± 6.18E+00†	6.63E−02 ± 2.48E−01
F10	1.76E−14 ± 4.21E−15†	4.94E−15 ± 1.57E−15≈	4.00E−15 ± 0.00E+00‡	2.13E+00 ± 1.17E+00†	5.54E−15 ± 1.76E−15
F11	1.23E−03 ± 3.23E−03†	0.00E+00 ± 0.00E+00≈	1.48E−03 ± 3.90E−03†	1.52E−01 ± 3.70E−01†	0.00E+00 ± 0.00E+00
F12	1.73E−02 ± 7.61E−02≈	1.57E−32 ± 5.47E−48≈	3.46E−03 ± 1.86E−02≈	2.19E+02 ± 9.13E+02†	1.57E−32 ± 5.47E−48
F13	3.66E−04 ± 1.97E−03≈	1.35E−32 ± 5.47E−48≈	1.10E−03 ± 3.30E−03†	3.44E+04 ± 1.14E+05†	1.35E−32 ± 5.47E−48
F14	7.58E−14 ± 2.68E−14†	0.00E+00 ± 0.00E+00≈	0.00E+00 ± 0.00E+00≈	1.15E+01 ± 2.50E+01†	0.00E+00 ± 0.00E+00
F15	1.11E+01 ± 1.54E+01†	3.32E+02 ± 9.10E+01†	4.45E−12 ± 6.12E−12‡	5.68E+00 ± 2.90E+01†	9.71E−11 ± 1.13E−10
F16	1.65E+07 ± 5.56E+06†	2.46E+07 ± 6.98E+06†	1.75E+05 ± 1.20E+05‡	2.99E+05 ± 1.37E+05≈	2.48E+05 ± 1.11E+05
F17	9.91E+02 ± 4.14E+02†	5.03E+03 ± 9.98E+02†	1.80E−03 ± 3.72E−03‡	4.88E+00 ± 8.28E+00†	2.10E−03 ± 3.97E−03
F18	3.15E+03 ± 5.04E+02†	4.20E+03 ± 4.72E+02†	3.91E+02 ± 2.55E+02†	2.87E+03 ± 6.39E+02†	2.67E+02 ± 2.34E+02
F19	6.59E+01 ± 4.24E+01†	4.79E+01 ± 2.42E+01†	3.99E−01 ± 1.20E+00†	1.80E+07 ± 4.07E+07†	2.27E−14 ± 2.78E−14
F20	2.14E−02 ± 1.91E−02†	6.53E−02 ± 3.13E−02†	6.07E−03 ± 9.13E−03≈	4.27E−01 ± 6.46E−01≈	2.55E−03 ± 5.74E−03
F21	2.10E+01 ± 3.91E−02≈	2.09E+01 ± 4.32E−02≈	2.09E+01 ± 4.53E−02≈	2.10E+01 ± 4.05E−02†	2.09E+01 ± 5.57E−02
F22	5.74E+00 ± 3.86E+00†	9.95E−02 ± 2.98E−01≈	2.84E+01 ± 7.73E+00†	2.78E+01 ± 8.52E+00†	3.32E−02 ± 1.79E−01
F23	8.95E+01 ± 1.48E+01†	1.32E+02 ± 1.01E+01†	1.51E+02 ± 5.82E+01†	4.28E+01 ± 1.25E+01‡	6.67E+01 ± 1.75E+01
$w/l/t$	16/3/4	9/4/10	13/5/5	18/1/4	–

according to the paired Wilcoxon signedrank test, since the bad results achieved by DE1 appeared as noises for several times in the whole 30 runs. For shifted and rotated functions, however, DE1 could hardly achieve promising solutions. It demonstrates that DE/best/1 is only suitable for solving unimodal and simple multimodal problems. Compared to DE2, RADE wins on nine functions but loses on four functions. It is impressive that DE2 performs similar as RADE on F06–F13. However, on the shifted and/or rotated functions F15–F20, RADE is better than DE2. The possible reason is that DE/rand/1 coupled with low *CR* value is suitable for solving simple multimodal functions, but not for the shifted and/or rotated functions. Although DE3 achieves the best performance on F15–F17, RADE performs better on 13 functions. Compared with its own results on the first thirteen functions, DE3 seems to perform better on the remaining ten CEC 2005 functions. It may imply that the DE/rand/1 strategy with a larger CR value may be more suitable for the shifted and/or rotated functions. The most commonly used DE scheme, DE4, is much worse than RADE, since RADE wins on 18 out of 23 functions but only loses on one function, F23.

Figure 4 shows the convergence curves of the four DE schemes and RADE on some selected functions. Moreover, the selection probability of each role is also illustrated in the right column of Fig. 4.

From the right column of Fig. 4, it can be observed that the selection probability of exploiters is much lower than that of the other two roles, and it decreases as the evolution proceeds. This meets our expectation, the selection probability of exploiters may be the lowest among the three roles, and decreases at the middle or late stages of evolution. Because the search region covered by the population becomes smaller and smaller as the evolution proceeds, and hence, different individuals tend to cluster around the best individual. This implies that the number of those individuals that have good fitness values and are also far away from the best individual would decrease significantly. For the followers and explorers, there is a trend that the selection probability of the explorers is slightly larger than that of the followers at the early

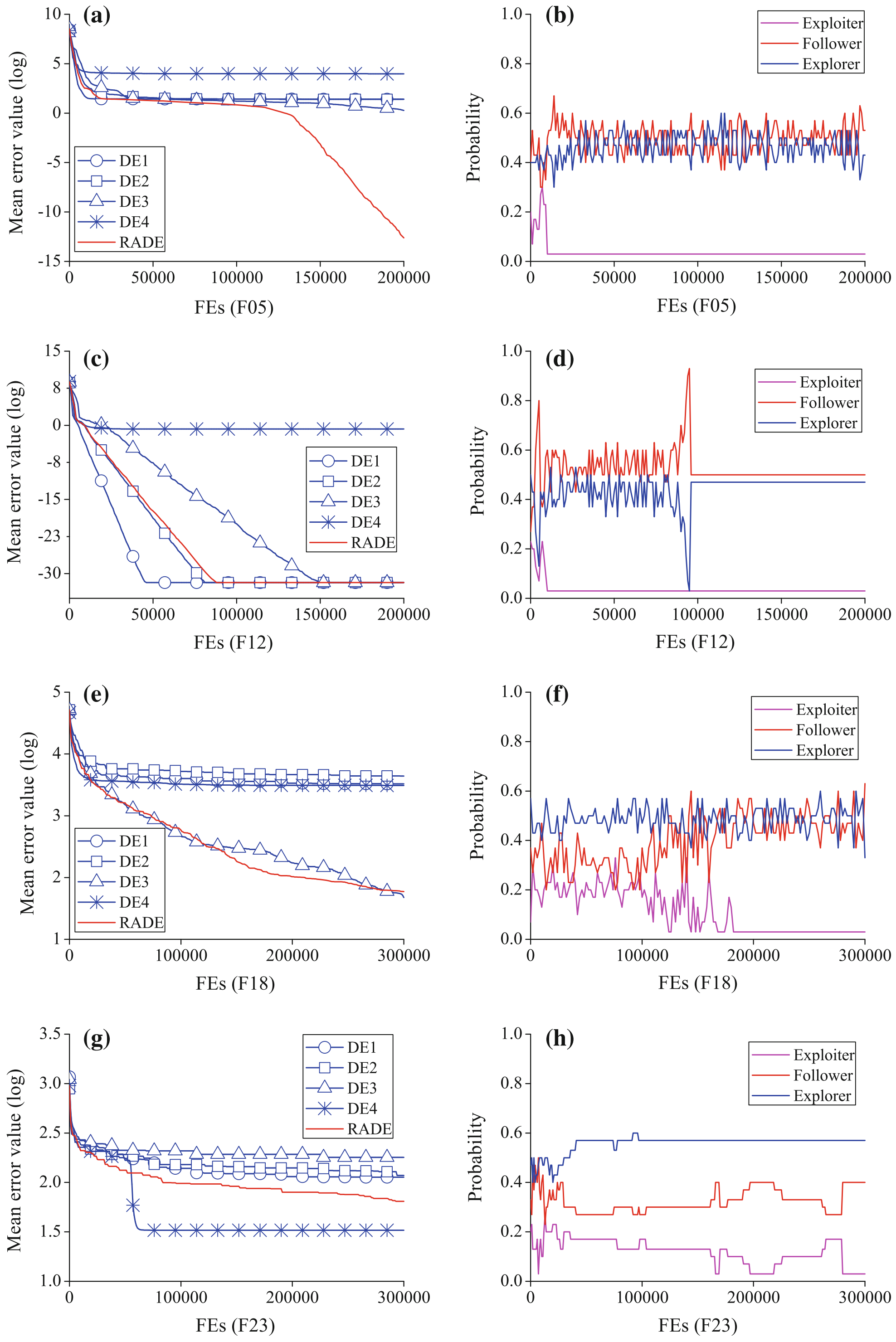

Fig. 4 The convergence curves of four DE schemes and RADE on four selected functions at $D = 30$ are illustrated in the *left column*, and the selection probability of each role is shown in the *right column*

stage of evolution, and then it becomes slightly lower than or similar to that of the followers, at the middle and late stages, see the F05 and F18. But there is an exception, on the F12, that the selection probability of followers is almost always larger than that of the explorers. The possible reason is that a greedy mutation strategy is more suitable for this function, thus exploitative search behavior is encouraged. This can also be proved by the corresponding convergence curves illustrated in the left column, where the DE/best/1/ strategy (namely DE1) achieves the best performance. In summary, based on the results and analysis, our approach is capable of providing more superior overall performance to the corresponding DE schemes. Our approach attempts to balance the local search ability of the exploiters and followers, and the global search ability of the explorers. In the next subsection, we will test the effect of dimensionality on our approach.

5.2.2 Scalability test

In order to better understanding of the performance, in this subsection, the scalability test is conducted. For functions F01–F13, the dimensions are scaled at $D = 50$, 100, and 200. While for functions F14–F23, only $D = 50$ is used, since these functions are defined up to $D = 50$ (Suganthan et al. (2005)). The results are respectively shown in the Tables 3, 4 and 5 for $D = 50$, 100 and 200. From the Table 3, it can be observed, although the complexity of a problem increases with its dimension, the performance of RADE is not always affected; instead, RADE consistently get significantly better results than its competitors. Specifically, RADE wins on 11 functions at $D = 50$ while on nine functions at $D = 30$, compared with DE2. For DE3 and DE4, RADE outperformed DE3 on 14 functions, and does not lose on any function compared with DE4.

For functions F01–F13 at $D = 100$ and $D = 200$, it is clear that RADE achieves the best performance among the compared DE schemes. It is impressive to see that the performance of DE2 is also very promising and competitive, since both DE2 and RADE achieve similar performance on several functions (F08–F13) when the dimension increases from 100 to 200. But considering the performance of RADE on the ten CEC 2005 functions when $D = 30$ and $D = 50$, we can draw the conclusion that RADE is better than DE2 as a whole.

5.2.3 Computational cost of RADE

In RADE, we adopt the RA scheme for each individual to select its mutation strategy and parameter settings. One important feature of the RA scheme is that the positional information of each individual is used, however this requires additional computational cost to calculate the dis-

Table 3 Comparisons of mean function error values and standard deviation for all test functions at $D = 50$

Funcion	DE1 Mean error ± standard deviation	DE2 Mean error ± standard deviation	DE3 Mean error ± standard deviation	DE4 Mean error ± standard deviation	RADE Mean error ± standard deviation
F01	1.19E−186 ± 0.00E+00‡	7.11E−106 ± 9.19E−106‡	2.14E−56 ± 8.91E−56†	8.75E+00 ± 3.09E+01†	1.71E−96 ± 3.50E−96
F02	1.47E−104 ± 2.10E−104‡	2.65E−61 ± 1.84E−61‡	3.97E−30 ± 5.26E−30†	2.63E−29 ± 1.41E−28†	4.91E−54 ± 3.77E−54
F03	1.72E+01 ± 2.71E+01†	7.98E+02 ± 1.46E+02†	4.84E−11 ± 8.44E−11‡	4.51E−02 ± 2.37E−01†	1.85E−10 ± 1.45E−10
F04	2.58E+01 ± 6.41E+00†	1.10E−11 ± 4.31E−12‡	3.89E+00 ± 3.06E+00†	2.05E+01 ± 4.33E+00†	1.95E−09 ± 8.38E−10
F05	4.04E+01 ± 2.92E+01†	3.09E+01 ± 1.57E+01†	6.75E+00 ± 2.03E+01†	6.47E+03 ± 1.66E+04†	1.32E−20 ± 6.47E−20
F06	6.67E−02 ± 2.49E−01 ≈	0.00E+00 ± 0.00E+00 ≈	0.00E+00 ± 0.00E+00 ≈	0.00E+00 ± 0.00E+00 ≈	0.00E+00 ± 0.00E+00
F07	2.15E−03 ± 7.04E−04‡	4.12E−03 ± 1.13E−03†	4.50E−03 ± 1.67E−03†	7.56E−03 ± 3.67E−03†	3.58E−03 ± 8.07E−04
F08	2.28E+02 ± 1.66E+02†	3.82E−04 ± 0.00E+00‡	9.04E+02 ± 4.13E+02†	1.30E+03 ± 4.65E+02†	3.95E+01 ± 5.58E+01
F09	2.39E+00 ± 1.81E+00†	3.32E−02 ± 1.79E−01 ≈	3.02E+01 ± 1.01E+01†	1.90E+01 ± 5.85E+00†	9.95E−02 ± 2.98E−01
F10	1.83E−14 ± 4.72E−15†	4.47E−15 ± 1.21E−15 ≈	4.00E−15 ± 0.00E+00‡	2.03E+00 ± 1.01E+00†	5.30E−15 ± 1.71E−15
F11	9.86E−04 ± 3.16E−03 ≈	0.00E+00 ± 0.00E+00 ≈	2.87E−03 ± 4.97E−03†	1.69E−01 ± 2.14E−01†	4.93E−04 ± 1.84E−03
F12	1.57E−32 ± 5.47E−48 ≈	1.57E−32 ± 5.47E−48 ≈	3.11E−02 ± 1.04E−01†	1.10E+03 ± 3.94E+03†	1.57E−32 ± 5.47E−48
F13	3.66E−04 ± 1.97E−03 ≈	1.35E−32 ± 5.47E−48 ≈	3.66E−04 ± 1.97E−03†	5.84E+04 ± 1.83E+05†	1.35E−32 ± 5.47E−48
F14	1.53E−13 ± 3.64E−14†	5.68E−14 ± 0.00E+00†	0.00E+00 ± 0.00E+00‡	2.44E+02 ± 7.40E+02†	4.17E−14 ± 2.51E−14
F15	1.68E+03 ± 6.33E+02†	1.30E+04 ± 1.90E+03†	5.33E−04 ± 6.54E−04‡	2.36E−02 ± 5.32E−02 ≈	3.34E−02 ± 3.72E−02
F16	5.40E+07 ± 1.53E+07†	8.84E+07 ± 1.83E+07†	4.86E+05 ± 1.57E+05‡	5.44E+05 ± 1.96E+05 ≈	7.33E+05 ± 2.76E+05
F17	1.82E+04 ± 4.54E+03†	3.61E+04 ± 4.43E+03†	3.34E+02 ± 3.06E+02 ≈	1.84E+03 ± 1.47E+03†	4.82E+02 ± 4.14E+02
F18	7.70E+03 ± 1.28E+03†	1.05E+04 ± 1.05E+03†	2.62E+03 ± 5.02E+02 ≈	6.94E+03 ± 1.20E+03†	2.38E+03 ± 4.10$E + 02$
F19	1.27E+02 ± 5.86E+01†	6.89E+01 ± 2.51E+01†	3.08E+01 ± 2.87E+01†	1.26E+08 ± 1.52E+08†	1.33E−01 ± 7.16E−01
F20	1.97E−02 ± 1.66E−02†	1.84E−02 ± 2.54E−02†	6.48E−03 ± 8.52E−03†	2.35E+00 ± 3.82E+00†	4.11E−04 ± 2.21E−03
F21	2.11E+01 ± 3.39E−02 ≈	2.11E+01 ± 3.31E−02 ≈	2.11E+01 ± 3.24E−02 ≈	2.11E+01 ± 3.62E−02 ≈	2.11E+01 ± 4.90E−02
F22	1.37E+01 ± 4.77E+00†	2.65E−01 ± 4.40E−01 ≈	6.54E+01 ± 1.16E+01†	7.96E+01 ± 1.42E+01†	6.30E−01 ± 1.01E+00
F23	2.36E+02 ± 2.39E+01†	3.37E+02 ± 2.42E+01†	3.45E+02 ± 8.82E+01†	1.07E+02 ± 2.11E+01 ≈	2.03E+02 ± 2.99E+01
$w/l/t$	15/3/5	11/4/8	14/5/4	18/0/5	–

Table 4 Comparisons of mean function error values and standard deviation for all test functions at $D = 100$

Function	DE1 Mean error ± standard deviation	DE2 Mean error ± standard deviation	DE3 Mean error ± standard deviation	DE4 Mean error ± standard deviation	RADE Mean error ± standard deviation
F01	0.00E+00 ± 0.00E+00‡	1.17E−215 ± 0.00E+00‡	1.49E−118 ± 4.11E−118†	2.28E+00 ± 6.66E+00†	1.13E−196 ± 0.00E+00
F02	7.03E−211 ± 0.00E+00‡	1.01E−123 ± 1.11E−123‡	8.20E−62 ± 1.54E−61†	1.08E−51 ± 5.81E−51≈	3.40E−109 ± 5.53E−109
F03	8.21E−01 ± 2.09E+00†	4.50E+01 ± 1.67E+01†	1.21E−25 ± 2.59E−25≈	1.27E−01 ± 5.84E−01†	4.09E−25 ± 8.82E−25
F04	2.50E+01 ± 5.80E+00†	1.79E−24 ± 7.82E−25‡	2.72E+00 ± 4.28E+00†	2.11E+01 ± 5.02E+00†	2.36E−19 ± 2.15E−19
F05	3.87E+01 ± 2.72E+01†	2.53E+01 ± 2.04E+00†	4.50E+00 ± 1.68E+01†	1.12E+04 ± 3.80E+04†	1.64E−31 ± 8.85E−31
F06	2.33E−01 ± 7.61E−01†	0.00E+00 ± 0.00E+00≈	0.0E+00 ± 0.00E+00≈	0.00E+00 ± 0.00E+00≈	0.00E+00 ± 0.00E+00
F07	1.14E−03 ± 2.82E−04‡	2.22E−03 ± 5.13E−04†	2.51E−03 ± 1.09E−03†	3.63E−03 ± 2.50E−03†	1.97E−03 ± 4.38E−04
F08	3.19E+02 ± 1.75E+02†	3.82E−04 ± 0.00E+00≈	7.07E+02 ± 3.71E+02†	1.33E+03 ± 5.30E+02†	7.90E+00 ± 2.95E+01
F09	3.05E+00 ± 2.19E+00†	3.32E−02 ± 1.82E−01≈	2.66E+01 ± 7.18E+00†	2.01E+01 ± 6.40E+00†	1.33E−01 ± 3.38E−01
F10	1.75E−14 ± 4.44E−15†	4.12E−15 ± 6.49E−16≈	4.00E−15 ± 0.00E+00≈	1.62E+00 ± 1.02E+00†	4.23E−15 ± 8.86E−16
F11	3.94E−03 ± 6.59E−03†	0.00E+00 ± 0.00E+00≈	1.89E−03 ± 3.85E−03†	2.82E−01 ± 7.03E−01†	2.47E−04 ± 1.33E−03
F12	1.57E−32 ± 5.47E−48≈	1.57E−32 ± 5.57E−48≈	4.19E−02 ± 2.25E−01≈	9.63E+02 ± 3.22E+03†	1.57E−32 ± 5.47E−48
F13	1.42E−32 ± 3.54E−33≈	1.35E−32 ± 0.00E+00≈	1.10E−03 ± 3.30E−03†	6.59E+04 ± 2.36E+05†	1.35E−32 ± 5.47E−48
$w/l/t$	8/3/2	3/3/7	9/0/4	11/0/2	–

Table 5 Comparisons of mean function error values and standard deviation for all test functions at $D = 200$

Funcion	DE1 Mean error ± standard deviation	DE2 Mean error ± standard deviation	DE3 Mean error ± standard deviation	DE4 Mean error ± standard deviation	RADE Mean error ± standard deviation
F01	0.00E+00 ± 0.00E+00≈	0.00E+00 ± 0.00E+00≈	8.02E−242 ± 0.00E+00†	2.38E+01 ± 5.02E+01†	0.00E+00 ± 0.00E+00
F02	4.90E−324 ± 0.00E+00‡	4.86E−249 ± 0.00E+00‡	2.65E−125 ± 1.12E−124†	3.91E−143 ± 2.10E−142†	1.82E−219 ± 0.00E+00
F03	1.47E−04 ± 7.84E−04†	1.50E−01 ± 9.94E−02†	1.59E−55 ± 7.21E−55≈	7.60E−04 ± 2.73E−03†	3.40E−54 ± 6.78E−54
F04	2.52E+01 ± 5.38E+00†	3.91E−50 ± 3.10E−50‡	9.49E−01 ± 1.92E+00†	2.05E+01 ± 6.75E+00†	1.31E−39 ± 1.74E−39
F05	3.21E+01 ± 2.43E+01†	2.65E+01 ± 1.04E+01†	4.50E+00 ± 1.68E+01†	3.54E+03 ± 1.15E+04†	0.00E+00 ± 0.00E+00
F06	3.33E−02 ± 1.80E−01≈	0.00E+00 ± 0.00E+00≈	0.00E+00 ± 0.00E+00≈	0.00E+00 ± 0.00E+00≈	0.00E+00 ± 0.00E+00
F07	6.46E−04 ± 1.79E−04≈	1.07E−03 ± 2.32E−04†	1.29E−03 ± 6.23E−04†	2.68E−03 ± 1.33E−03†	8.75E−04 ± 1.93E−04
F08	3.18E+02 ± 2.42E+02†	3.82E−04 ± 0.00E+00≈	7.36E+02 ± 3.92E+02†	1.34E+03 ± 4.27E+02†	3.55E+01 ± 6.94E+01
F09	2.89E+00 ± 1.57E+00†	9.95E−02 ± 2.98E−01≈	3.04E+01 ± 9.31E+00†	1.96E+01 ± 5.42E+00†	1.33E−01 ± 3.38E−01
F10	1.88E−14 ± 4.95E−15†	4.00E−15 ± 0.00E+00≈	4.00E−15 ± 0.00E+00≈	1.72E+00 ± 7.83E−01†	4.00E−15 ± 0.00E+00
F11	1.73E−03 ± 3.93E−03†	0.00E+00 ± 0.00E+00≈	2.87E−03 ± 4.72E−03†	1.47E−01 ± 3.76E−01†	0.00E+00 ± 0.00E+00
F12	1.59E−32 ± 9.27E−34≈	1.57E−32 ± 5.47E−48≈	1.57E−32 ± 5.47E−48≈	1.06E+02 ± 5.53E+02†	1.57E−32 ± 5.47E−48
F13	7.32E−04 ± 2.74E−03†	1.35E−32 ± 5.47E−48≈	7.32E−04 ± 2.74E−03≈	4.62E+04 ± 1.50E+05†	1.35E−32 ± 5.47E−48
$w/l/t$	8/1/4	3/2/8	8/0/5	12/0/1	–

tances between X_i ($i = 1, 2, \ldots, NP$) and X_b. Obviously, the major difference of computational cost between basic DE and our approach is the overhead of Eq. (10), which has to calculate $(NP - 1)$ number of distances at each generation. Strictly speaking, for a specific optimization problem of D-dimension, the computational cost of an evolutionary algorithm is not only affected by its evolutionary operations, but also by the evaluation of the objective function (Epitropakis et al. 2011). Assume that the computational cost of an objective function evaluation be equal to c units of real computational time, while the cost of computing a distance between two individuals be $(\lambda \cdot c)$ units of real computational time. Take the DE/rand/1 scheme as the representative basic DE, its computational cost per generation is $Cost_{DE} = NP \cdot c$, while that of RADE is $Cost_{RADE} = (NP-1)\cdot\lambda\cdot c+NP\cdot c$. We would like to point out that the value of λ is relative to the complexity of the objective function. Generally, as the complexity increases, the ratio of the computational cost of RADE to that of basic DE would comparatively decrease. One can evaluate the ratio $Cost_{RADE}/Cost_{DE}$ to obtain an estimate of the computational overhead of our approach.

Table 6 The average CPU time (in seconds) of DE/rand/1 and RADE on the test suite

Funcion	$D = 30$			$D = 50$		
	DE/rand/1	RADE	Ratio	DE/rand/1	RADE	Ratio
F01	0.58	0.80	1.36	1.05	1.42	1.35
F02	0.59	0.80	1.35	1.03	1.25	1.21
F03	0.69	0.93	1.36	1.45	1.89	1.30
F04	0.57	0.76	1.34	1.00	1.32	1.32
F05	0.86	1.02	1.19	1.58	1.97	1.25
F06	0.69	0.85	1.23	1.32	1.54	1.17
F07	2.41	2.60	1.08	4.87	5.05	1.04
F08	1.13	1.33	1.18	2.24	2.64	1.18
F09	0.97	1.08	1.11	1.96	2.11	1.07
F10	0.91	1.03	1.14	1.95	2.01	1.03
F11	1.08	1.26	1.17	1.99	2.31	1.16
F12	1.42	1.47	1.03	3.00	2.98	0.99
F13	1.51	1.42	0.94	3.31	3.03	0.92
F14	0.94	1.25	1.33	2.16	2.76	1.28
F15	0.93	1.26	1.36	2.09	2.81	1.34
F16	4.23	4.53	1.07	11.84	12.10	1.02
F17	0.96	1.30	1.36	2.14	3.14	1.47
F18	1.29	1.63	1.27	3.97	4.85	1.22
F19	0.92	1.24	1.35	2.18	2.71	1.24
F20	2.21	2.43	1.10	6.70	7.04	1.05
F21	2.65	3.04	1.15	7.47	8.50	1.14
F22	1.64	1.68	1.02	3.96	4.04	1.02
F23	2.36	2.77	1.17	6.74	7.74	1.15
Overall	31.53	36.49	1.16	75.98	85.21	1.12

For a quantified comparison, we employ the test suite used in the experiments to investigate the computational runtimes of both RADE and basic DE. Table 6 represents the CPU time of DE/rand/1 and RADE on the test suite. Each algorithm is run 30 times per function, and the average CPU time is recorded. The used stopping criterion is the same as before. The computational configurations are listed as follows.

- OS: Windows 7 (x64).
- CPU: Intel Core 2 Quad CPU Q8200 (2.33 GHz).
- RAM: 4G.
- Language: Java.
- Compiler: Eclipse SDK 4.2.0.

From Table 6, we can see that when $D = 30$, the range of the ratios on different functions is [0.94, 1.36] and the overall ratio is 1.16. The highest ratio is on F01, while the lowest one is on F13. It is interesting to note that the ratio on F13 is even less than 1, which implies that the computational cost of RADE is smaller than DE/rand/1 on this function. The possible reasons are as follows. First, the simple Sphere function (F01) only includes the calculation of the sum of squares, while the Penalized function (F13) is much harder to calculate, thus the computational costs of both algorithms almost spend on the calculation of the objective function rather than on the evolutionary operations. Second, since DE is an evolutionary algorithm, the stochastic characteristic of such algorithms is also capable of affecting the cost. Therefore, for a relatively hard problem, RADE has the chance to solve it with less time in comparison with basic DE. When $D = 50$, the overall ratio is 1.12 which is less than 1.16 of $D = 30$, and the ratios on most functions decrease as well, because the complexity of a problem increases with the dimension. Furthermore, we can conclude that the ratio on a multimodal function is usually less than on a unimodal function, which implies that as the complexity increases, the computational cost of RADE would comparatively decrease.

Table 7 Parameter settings for the five competitive DE variants

Algorithms	Parameter settings
jDE	$NP = 100,\ \tau_1 = 0.1,\ \tau_2 = 0.1$
SaDE	$NP = 50,\ LP = 50$
ODE	$NP = 100,\ F = 0.5,\ CR = 0.9,\ J_r = 0.3$
EPSDE	$NP = 50$
MGBDE	$NP = 100,\ F = 0.5,\ CR = 0.9$

5.3 Comparison of RADE with some state-of-the-art DE variants

In this section, we compare RADE with five other recently proposed DE variants, which are listed as follows.

(a) self-adapting DE (jDE) (Brest et al. 2006);
(b) self-adaptive DE (SaDE) (Qin et al. 2009);
(c) opposition-based DE (ODE) (Rahnamayan et al. 2008);
(d) DE with ensemble of parameters and mutation strategies (EPSDE) (Mallipeddi et al. 2011);
(e) modified Gaussian bare-bone DE (MGBDE) (Wang et al. 2013).

The control parameters for these five competitive algorithms are set the same as in the original papers, and the parameters without adaptive adjustments are listed in Table 7. For those adaptively adjusted parameters of each competitive algorithm (except for ODE, since all of its parameters are fixed), they are described as follows. In jDE, the ranges of F and CR values are [0.1, 1.0] and [0, 1], respectively. Both F and CR are adapted based on two constants τ_1 and τ_2. In SaDE, the F values are randomly sampled from a normal distribution $N(0.5, 0.3)$, and so the available F values are almost within $[-0.4, 1.4]$. Compared with F, the adaptation for CR

Table 8 Comparisons of RADE with five other DE variants for all functions at $D = 30$

Function	jDE (mean)	SaDE (mean)	ODE (mean)	EPSDE (mean)	MGBDE (mean)	RADE (mean)
F01	6.33E−40†	1.63E−87‡	3.01E+02†	3.42E−84‡	3.75E−50†	6.64E−76
F02	5.95E−24†	1.09E−52‡	2.01E−01†	2.97E−39†	2.90E−30†	5.19E−43
F03	6.82E−04†	3.37E−04†	5.97E+02†	7.06E−02†	4.64E+01†	1.94E−07
F04	8.07E−01†	2.22E−07≈	8.36E+00†	2.34E+00†	7.96E−06†	2.58E−07
F05	1.77E+01†	2.72E+01†	2.74E+04†	7.97E−01†	2.58E+01†	4.42E−11
F06	0.00E+00≈	0.00E+00≈	2.08E+02†	1.67E−01†	0.00E+00≈	0.00E+00
F07	5.38E−03≈	4.49E−03≈	4.91E−02†	2.11E−03‡	3.51E−03‡	4.60E−03
F08	3.82E−04‡	3.82E−04‡	2.64E+03†	3.82E−04‡	6.34E+02†	3.16E+01
F09	0.00E+00≈	3.32E−02≈	4.97E+01†	0.00E+00≈	5.14E+00†	6.63E−02
F10	5.86E−15†	9.31E−02≈	1.64E+00†	6.22E−15†	7.08E−15†	5.54E−15
F11	0.00E+00≈	2.14E−03†	1.26E−01†	1.48E−03†	7.40E−04≈	0.00E+00
F12	1.57E−32≈	1.57E−32≈	4.46E+03†	3.46E−03†	1.57E−32≈	1.57E−32
F13	1.35E−32≈	3.66E−04†	2.67E+03†	1.35E−32≈	1.35E−32≈	1.35E−32
F14	0.00E+00≈	1.68E−30≈	5.34E+01†	0.00E+00≈	4.93E−14†	0.00E+00
F15	2.03E−06†	6.52E−06†	1.95E+03†	6.25E−26‡	4.09E+00†	9.71E−11
F16	1.84E+05‡	4.69E+05†	6.52E+06†	5.83E+05†	6.01E+06†	2.48E+05
F17	7.12E−02†	9.10E+01†	2.28E+01†	2.13E+01†	1.65E+02†	2.10E−03
F18	4.32E+02†	3.24E+03†	3.65E+03†	1.64E+03†	1.86E+03†	2.67E+02
F19	2.36E+01†	3.75E+01†	5.91E+07†	3.99E−01≈	3.08E+01†	2.27E−14
F20	3.20E−03†	1.21E−02†	2.55E−02†	1.83E−02†	1.05E−02†	2.55E−03
F21	2.09E+01≈	2.09E+01≈	2.10E+01≈	2.09E+01≈	2.10E+01≈	2.09E+01
F22	0.00E+00≈	1.99E−01†	4.60E+01†	0.00E+00≈	5.70E+00†	3.32E−02
F23	5.57E+01‡	4.65E+01‡	8.05E+01≈	4.59E+01‡	1.65E+02†	6.67E+01
$w/l/t$	11/3/9	11/4/8	21/0/2	12/5/6	17/1/5	–

is more complicated. Similar to *F*, the *CR* values are also generated by a normal distribution $N(CR_m, Std)$, where Std is set to 0.1 while CR_m is initialized as 0.5 and then adjusted with previous experiences. In EPSDE, two parameter pools are defined for *F* and *CR*. The pool of *F* values is taken in the range [0.4, 0.9] in steps of 0.1, and the pool of *CR* values is taken in the range [0.1, 0.9] in steps of 0.1. In MGBDE, a simple but efficient self-adaptive mechanism is proposed to dynamically update *CR* for the Gaussian mutation strategy. A normal distribution $N(0.5, 0.1)$ is used to generate the *CR* values. If an individual with the current *CR* value cannot generate a better candidate solution, then the *CR* value will be updated by the same distribution in the next generation.

For RADE, we use the identical parameter settings as in previous Sect. 5.2. The current experiments are respectively conducted for $D = 30$ and $D = 50$. For $D = 30$, *MaxFEs* is set to 200,000 for the first 13 functions; but for the remaining ones, *MaxFEs* is equal to 300,000; and for $D = 50$, *MaxFEs* is set to 250,000 for the first 13 functions; but *MaxFEs* is set to 500,000 for the remaining functions. Each algorithm is run 30 times per function, and the mean function error values are recorded.

Tables 8 and 9 show the results at $D = 30$ and $D = 50$, respectively. "Mean" represents the mean function error value. From the results, it can be seen that our approach, RADE, achieves better results than its five competitors on the majority of test functions. To be specific, when $D = 30$ RADE outperforms jDE on 11 functions, but only loses on three functions. SaDE performs better than RADE on four functions, while RADE wins on 11 functions. ODE does not achieve better results than RADE on any functions, while RADE outperforms ODE on 21 functions. It is impressive that EPSDE performs best on the function F15, and the result is much better than other DE variants. But for other functions, RADE achieves better results than EPSDE on 12 functions out of them. MGBDE is only better than RADE on the function F07, while RADE wins on 17 functions. For $D = 50$, we also get the similar comparison results as $D = 30$. Figure 5 shows the convergence curves of the five competitive DE variants and RADE on four selected functions at $D = 30$.

In order to compare the performance of multiple algorithms on the test suite, we conducted the Friedman test (Garca et al. 2009, 2010). Tables 10 and 11 show the average rankings of the six DE algorithms for $D = 30$ and $D = 50$. The best ranking is shown in boldface. As seen, the best average ranking was obtained by the RADE algorithm, which outperforms the other five algorithms, the performance of the rest of algorithms can be sorted by average ranking

Table 9 Comparisons of RADE with five other DE variants for all functions at $D = 50$

Function	jDE (mean)	SaDE (mean)	ODE (mean)	EPSDE (mean)	MGBDE (mean)	RADE (mean)
F01	5.21E−35†	2.38E−58†	6.22E+00†	9.57E−92†	1.18E−33†	1.71E−96
F02	4.33E−21†	1.40E−39†	5.80E−01†	8.31E−52†	3.31E−22†	4.91E−54
F03	4.19E+00†	1.81E+00†	2.26E+02†	2.02E+03†	5.96E+03†	1.85E−10
F04	8.60E+00†	7.45E−02†	7.29E+00†	1.07E+01†	1.78E−01†	1.95E−09
F05	4.44E+01†	7.57E+01†	5.67E+05†	7.11E+00†	5.37E+01†	1.32E−20
F06	0.00E+00≈	0.00E+00≈	2.91E+02†	1.30E+00†	0.00E+00≈	0.00E+00
F07	1.00E−02†	1.74E−02†	4.67E−02†	6.47E−03†	9.74E−03†	3.58E−03
F08	6.36E−04‡	6.36E−04‡	2.05E+03†	6.36E−04‡	2.06E+03†	3.95E+01
F09	0.00E+00≈	4.31E−01†	5.31E+01†	6.63E−02≈	2.38E+01†	9.95E−02
F10	7.99E−15†	1.15E+00†	1.98E+00†	2.93E−02†	1.30E−14†	5.30E−15
F11	0.00E+00≈	1.54E−02†	2.94E−01†	3.77E−03†	0.00E+00≈	4.93E−04
F12	9.42E−33‡	2.91E−02†	1.90E+00†	4.15E−03†	6.22E−03†	1.57E−32
F13	1.35E−32≈	1.21E−01†	3.72E+03†	1.10E−03†	1.66E−32†	1.35E−32
F14	1.68E−30‡	5.05E−30‡	2.44E+03†	2.40E−29‡	5.68E−14†	4.17E−14
F15	1.24E−02≈	8.72E−02†	8.04E+03†	1.09E+03†	1.63E+03†	3.34E−02
F16	5.54E+05‡	9.75E+05†	1.46E+07†	9.39E+06†	1.26E+07†	7.33E+05
F17	4.65E+02≈	5.65E+03†	1.06E+03†	7.18E+03†	1.02E+04†	4.82E+02
F18	3.38E+03†	8.32E+03†	1.12E+04†	4.53E+03†	4.42E+03†	2.38E+03
F19	3.88E+01†	9.87E+01†	5.90E+08†	1.46E+00≈	4.66E+01†	1.33E−01
F20	4.27E−03†	8.52E−03†	3.82E+00†	9.92E−03≈	5.00E−03†	4.11E−04
F21	2.11E+01≈	2.11E+01≈	2.11E+01≈	2.11E+01≈	2.11E+01≈	2.11E+01
F22	0.00E+00‡	2.35E+00†	1.09E+02†	3.32E−02‡	2.65E+01†	6.30E−01
F23	9.61E+01‡	1.31E+02‡	2.17E+02≈	1.59E+02‡	3.55E+02†	2.03E+02
$w/l/t$	10/6/7	18/3/2	21/0/2	15/4/4	20/0/3	–

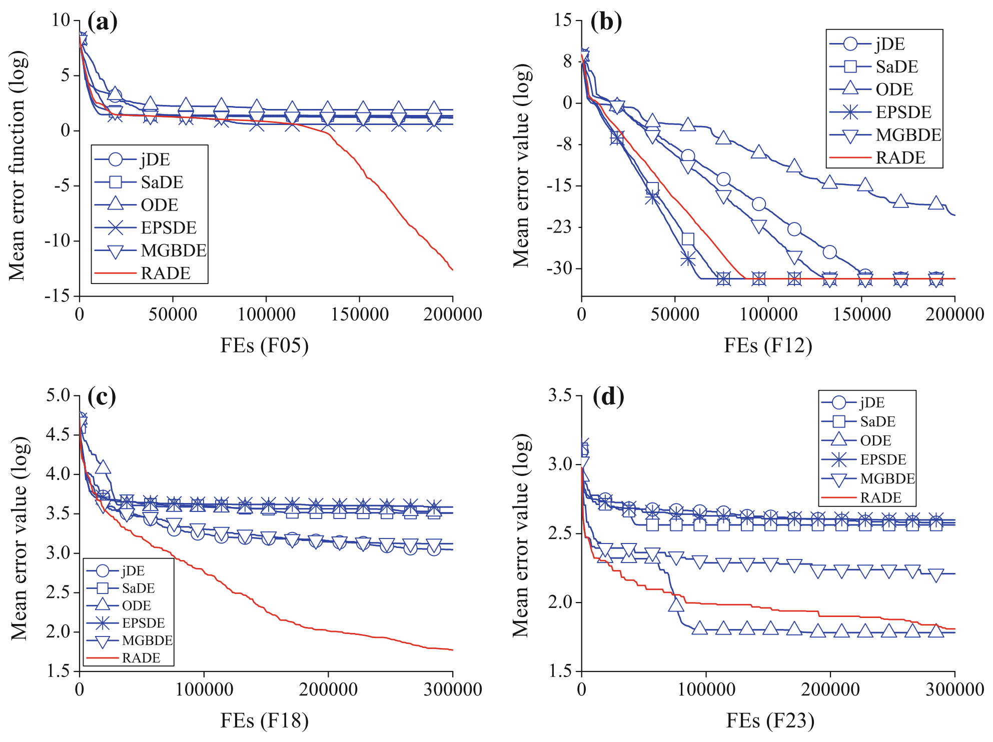

Fig. 5 The convergence curves of the five competitive DE variants and RADE on four selected functions at $D = 30$

Table 10 Average rankings achieved by Friedman test at $D = 30$

Algorithms	Average rankings
RADE	**2.20**
jDE	2.67
EPSDE	2.83
SaDE	3.33
MGBDE	4.13
ODE	5.8

Table 11 Average rankings achieved by Friedman test at $D = 50$

Algorithms	Average rankings
RADE	**2.02**
jDE	2.26
EPSDE	3.14
SaDE	3.65
MGBDE	4.15
ODE	5.50

into the following order: jDE, EPSDE, SaDE, MGBDE, and ODE.

6 Conclusions

DE is a popular population-based optimization algorithm, however its performance is very sensitive to the mutation strategy and control parameters. In this paper, we developed a RA scheme to dynamically select mutation strategies and parameter settings for different individuals of the population. The RA scheme utilizes both the fitness information and positional information of individuals to dynamically divide the population into three groups. Each group is considered as a role to exhibit different search behavior. There are three roles in total, i.e. *exploiter*, *follower* and *explorer*. Each role has its own mutation strategy and parameter setting during the evolution. The structure of our approach, RADE, is simple and it is easy to implement.

The experimental studies in this paper were carried out on 23 well-known benchmark problems including shifted and rotated functions. The proposed approach RADE is compared with four different DE schemes, and five recently proposed DE variants on several scale of dimensionality. The experimental results suggested that its overall performance was better than, or at least comparable to, its competitors. In the future, it is very interesting to study whether the RA scheme can be used to enhance other EAs' performance, such as particle swarm optimization (PSO) algorithm and artificial bee colony (ABC) algorithm.

Acknowledgments This work is supported by the National Natural Science Foundation of China (No. 61070008, 61364025), the Foundation of State Key Laboratory of Software Engineering (No. SKLSE2012-09-19), and the Fundamental Research Funds for the Central Universities (No. 2012211020205).

References

Abbass HA (2002) The self-adaptive pareto differential evolution algorithm. In: IEEE conference on evolutionary computation, vol 1, pp 831–836

Brest J, Greiner S, Boskovic B, Mernik M, Zumer V (2006) Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. IEEE Trans Evol Comput 10(6):646–657

Caponio A, Neri F, Tirronen V (2009) Super-fit control adaptation in memetic differential evolution frameworks. Soft Comput 13(8–9):811–831

Das S, Suganthan PN (2010) Differential evolution: a survey of the state-of-the-art. IEEE Trans Evol Comput 15(1):4–31

Das S, Abraham A, Konar A (2008) Automatic clustering using an improved differential evolution algorithm. IEEE Trans Syst Man Cybern Part A Syst Hum 38(1):218–237

Eiben AE, Hinterding R, Michalewicz Z (1999) Parameter control in evolutionary algorithms. IEEE Trans Evol Comput 3(2):124–141

Epitropakis MG, Tasoulis DK, Pavlidis NG, Plagianakos VP, Vrahatis MN (2011) Enhancing differential evolution utilizing proximity-based mutation operators. IEEE Trans Evol Comput 15(1):99–119

Garca S, Molina D, Lozano M, Herrera F (2009) A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the cec' 2005 special session on real parameter optimization. J Heuristics 15(6):617–644

Garca S, Fernndez A, Luengo J, Herrera F (2010) Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power. Inf Sci 180(10):2044–2064

Ghosh A, Chowdhury A, Giri R (2010) A fitness-based adaptation scheme for control parameters in differential evolution. In: Genetetic and evolutionary computation conference, pp 2075–2076

Ghosh A, Das S, Chowdhury A, Giri R (2011) An improved differential evolution algorithm with fitness-based adaptation of the control parameters. Inf Sci 181(18):3749–3765

Gong W, Cai Z (2013) Differential evolution with ranking-based mutation operators. IEEE Trans Cybern. doi:10.1109/TCYB.2013.2239988

Gong W, Cai Z, Ling CX (2011a) DE/BBO: a hybrid differential evolution with biogeography-based optimization for global numerical optimization. Soft Comput 15(4):645–665

Gong W, Fialho Cai Z (2011b) Adaptive strategy selection in differential evolution for numerical optimization: an empirical study. Inf Sci 181:53645386

Liu J, Lampinen J (2005) A fuzzy adaptive differential evolution algorithm. Soft Comput 9(6):448–462

Mallipeddi R, Suganthan PN, Pan QK, Tasgetiren MF (2011) Differential evolution algorithm with ensemble of parameters and mutation strategies. Appl Soft Comput 11(2):1679–1696

Maulik U, Saha I (2010) Automatic fuzzy clustering using modified differential evolution for image classification. IEEE Trans Geosci Remote Sens 48(9):3503–3510

Neri F, Tirronen V (2010) Recent advances in differential evolution: a survey and experimental analysis. Artif Intell Rev 33(1):61–106

Noman N, Iba H (2008) Accelerating differential evolution using an adaptive local search. IEEE Trans Evol Comput 12(1):107–125

Price K, Storn R, Lampinen J (2005) Differential evolution: a practical approach to global optimization. Springer-Verlag, New York

Qin AK, Suganthan PN (2005) Self-adaptive differential evolution algorithm for numerical optimization. In: IEEE congress on evolutionary computation, vol 2, pp 1785–1791

Qin AK,LHuang V, Suganthan PN(2009) Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Trans Evol Comput 13(2):398–417

Rahnamayan S, Tizhoosh HR, Salama MMA (2008) Opposition-based differential evolution. IEEE Trans Evol Comput 12(1):64–79

Ronkkonen J, Kukkonen S, Price KV (2005) Real-parameter optimization with differential evolution. In: IEEE congress on evolutionary computation, vol 1, pp 506–513

Salman A, Engelbrecht AP, Omran MGH (2007) Empirical analysis of self-adaptive differential evolution. Eur J Oper Res 183(2):785–804

Shang YW, Qiu YH (2006) A note on the extended rosenbrock function. Evol Comput 14(1):119–126

Storn R, Price K (1997) Differential evolutional simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim 11(4):341–359

Suganthan PN, Hansen N, Liang JJ, Deb K, Chen YP, Auger A, Tiwari S (2005) Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization. In: Technical report, Nanyang Technological University, Singapore

Sun J, Zhang Q, KTsang EP, (2005) DE/EDA: a new evolutionary algorithm for global optimization. Inf Sci 169(3–4):249–262

Wang H, Wu Z, Rahnamayan S (2011a) Enhanced opposition-based differential evolution for solving high-dimensional continuous optimization problems. Soft Comput 1–14

Wang Y, Cai Z, Zhang Q (2011b) Differential evolution with composite trial vector generation strategies and control parameters. IEEE Trans Evol Comput 15(1):55–66

Wang Y, Cai Z, Zhang Q (2012) Enhancing the search ability of differential evolution through orthogonal crossover. Inf Sci 185(1):153–177

Wang H, Rahnamayan S, Sun H, Omran MGH (2013) Gaussian barebones differential evolution. IEEE Trans Cybern 43(2):634–647

Yang Z, Tang K, Yao X (2008) Large scale evolutionary optimization using cooperative coevolution. Inf Sci 178(15):2985–2999

Yao X, Liu Y, Lin G (1999) Evolutionary programming made faster. IEEE Trans Evol Comput 3(2):82–102

Zaharie D (2009) Influence of crossover on the behavior of differential evolution algorithms. Appl Soft Comput J 9(3):1126–1138

Zhang J, Sanderson AC (2009) JADE: adaptive differential evolution with optional external archive. IEEE Trans Evol Comput 13(5):945–958