

# APDDE: self-adaptive parameter dynamics differential evolution algorithm

Hong-bo Wang<sup>1,2</sup> · Xue-na Ren<sup>1,2</sup> · Guo-qing Li<sup>1,2</sup> · Xu-yan Tu<sup>1,2</sup>

Published online: 11 November 2016  
© Springer-Verlag Berlin Heidelberg 2016

**Abstract** In real-time high-dimensional optimization problem, how to quickly find the optimal solution and give a timely response or decisive adjustment is very important. This paper suggests a self-adaptive differential evolution algorithm (abbreviation for APDDE), which introduces the corresponding detecting values (the values near the current parameter) for individual iteration during the differential evolution. Then, integrating the detecting values into two mutation strategies to produce offspring population and the corresponding parameter values of champion are retained. In addition, the whole populations are divided into a pre-defined number of groups. The individuals of each group are attracted by the best vector of their own group and implemented a new mutation strategy *DE/Current-to-lbest/l* to keep balance of exploitation and exploration capabilities during the differential evolution. The proposed variant, APDDE, is examined on several widely used benchmark functions in the CEC 2015 Competition on Learning-based Real-Parameter Single Objective Optimization (13 global numerical optimization problems) and 7 well-known basic benchmark functions, and the experimental results show that the proposed APDDE algorithm improves the existing performance of other algorithms when dealing with the high-dimensional and multimodal problems.

**Keywords** Differential evolution · Self-adapting strategy · Real-time optimization

## 1 Introduction

DE (differential evolution) was presented at the Second International Contest on Evolutionary Optimization in 1997, and it turned out as one of the best among the competing algorithms (Storn and Price 1997; Price 1997). Since late 1990s, DE started to find several significant applications to the optimization problems. There are many reasons why the researchers have been looking at DE as an attractive optimization tool. For instance, it is simple and straight forward to implement, and the number of control parameters in DE is very few, and the space complexity of DE is low (Das and Suganthan 2011). Its performance, however, is still quite having some limitations; if DE cannot produce offspring better than the parent, the algorithm will be to a standstill. The reason is that: In every stage of the optimization process, the mobile is limited. Obviously, the differential evolution algorithm to successfully solve a specific optimization problem depends on the results of the mutation strategy and the three control parameters: the population size (NP), crossover rate (CR) and the scale factor ( $F$ ). The best settings for the control parameters can be different according to those practical optimization problems and the same functions with different requirements for consumption time and accuracy. Otherwise, the mobile restrictions will be to a standstill. Therefore, researchers have developed some DE variants with adaptive control parameters and trial vector generation strategies to suit various requirements during the evolution process. In the DE family of Storn and Price, *DE/rand/l/bin* is the most frequently used mutation strategies and many algorithms are developed based on it for its robustness and

Communicated by V. Loia.

✉ Hong-bo Wang  
foreverwhb@ustb.edu.cn

<sup>1</sup> Department of Computer Science and Technology, School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing 100083, People's Republic of China

<sup>2</sup> Beijing Key Laboratory of Knowledge Engineering for Materials Science, No. 30 Xueyuan Road, Haidian Zone, Beijing 100083, People's Republic of China

efficiency in terms of convergence rate. And *DE/rand/2/dir* (Feoktistov and Janaqi 2004) implements greedy DE strategies to generate offspring. However, a greedy DE variant such as *DE/current-to-best/1/bin* which employs the information of the best solution in the current population is less useful or difficultly to be developed. The reason seems to be very intuitive: less reliable and may lead to premature convergence. But Das and Suganthan (2011) noted that for non-separable and unimodal functions, *DE/current-to-best/1/bin* consistently showed the best phenomenon and it was also successful in optimizing some multimodal and separable benchmarks. Hence, these greedy DE variant strategies have great development of space; Zhang and Sanderson (2009) proposed a new algorithm which implements a new mutation strategy *DE/current-to-pbest* with optional external archive and achieved considerably good results.

The remaining part of this paper is organized in the following manner. In Sect. 2, we introduce the standard DE algorithm. Sect. 3 gives a detailed description about some recent variants that are commonly considered as the state of the art, such as JADE (Zhang and Sanderson 2009), MDE (Ali et al. 2013), CODE (Wang et al. 2011), MDESELF (Li and Yin 2014) and EPSDE (Mallipeddi et al. 2011). Section 4 presents the proposed APDDE algorithm. Section 5 makes comparative experiments on the CEC 2015 test suite and analyzes the related experimental results. Conclusions are drawn in Sect. 6.

## 2 Basic operations of DE

DE is a simple real-coded evolutionary algorithm. It works through a simple cycle of stages. The basic idea of the algorithm is: DE perturbs current generation vectors with the scaled difference in two randomly selected population vectors. The result is to generate a *donor vector*; it corresponds to each population vector (also known as *target vector*). Then, the donor vector exchanges its components with the target vector under crossover operation to produce a *trial vector*. In selection stage, the trial (or offspring) vector competes against the population vector; the better one will be used as the parent vector in the next generation. Hence, the population either gets better or remains current status, but never deteriorates, guiding the population to tend to the optimal solution.

Stages of DE presented in Fig. 1 includes: (1) initial parameter population, (2) mutation with *difference vec-*

*tors* to produce the *donor vector*, (3) setting the violating components within the predefined boundary constraints, (4) crossover operation between the target vector and its corresponding donor vector, (5) selecting operation through the objective function value of each trial vector compared with that of its corresponding target vector, and so on.

### 2.1 Initialization of the parameter vectors

DE searches for a global optimum parallelly in  $D$ -dimensional real-parameter space. It begins with an initiated population of NP with  $D$ -dimensional real-valued parameter vectors, which should cover the entire search space as much as possible by uniformly randomizing the initial individuals within the prescribed search space. For example, according to a uniform distribution:

$$x_{j,i}^{\text{Low}} \leq x_{j,i}(0) \leq x_{j,i}^{\text{Up}}, i = 1, 2, \dots, \text{NP}; j = 1, 2, \dots, D,$$

The initial value of the  $j$ th parameter of the  $i$ th individual at generation  $g = 0$  is generated by:

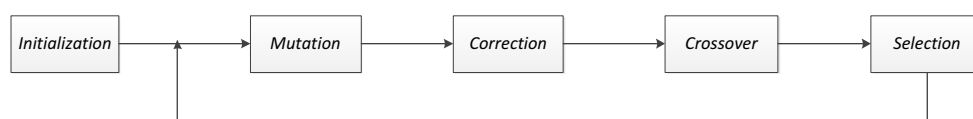
$$x_{j,i}(0) = x_{j,i}^{\text{Low}} + \text{rand}(0, 1) \cdot (x_{j,i}^{\text{Up}} - x_{j,i}^{\text{Low}}), \quad (1)$$

where  $x_{j,i}^{\text{Low}}, x_{j,i}^{\text{Up}}$  is the prescribed minimum and maximum bounds,  $D$  is the dimension of the problem, NP is the population size, and  $\text{rand}(0, 1)$  is a uniformly distributed random number between 0 and 1 and is instantiated independently for each component of the  $i$ th vector.

### 2.2 Mutation with difference vectors

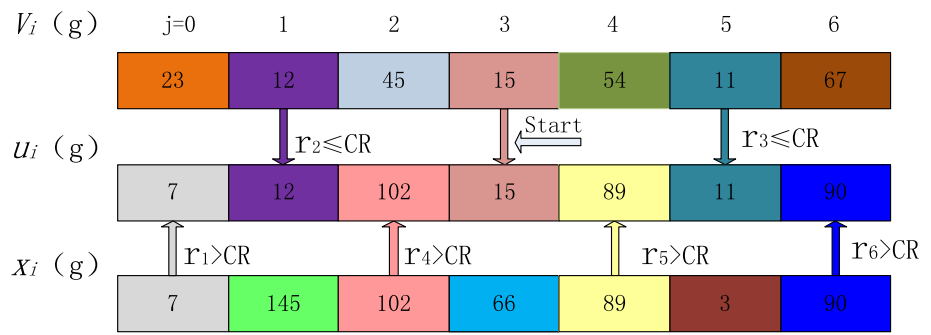
During DE-mutation of *DE/rand/1*, in order to produce the donor vector for each  $i$ th target vector from the current population, three other distinct parameter vectors,  $x_{r_1}(g), x_{r_2}(g), x_{r_3}(g)$  are sampled randomly from the current population. The indices  $r_1, r_2$  and  $r_3$  are distinct integers uniformly chosen from the set  $\{1, 2, \dots, \text{NP}\} \setminus \{i\}$ . These indices are randomly generated once for each mutant vector. The difference in any two of these three vectors is scaled by a scalar number  $F$  and added to the third one. So, the donor vector  $v_i(g)$  can be expressed as formula (2).

$$v_i(g) = x_{r_1}(g) + F \cdot (x_{r_2}(g) - x_{r_3}(g)). \quad (2)$$



**Fig. 1** Main stages of the DE algorithm

**Fig. 2** Crossover operation process



The following are other different mutation strategies frequently used in the literature:

1. *DE/rand/2*

$$v_i(g) = x_{r1}(g) + F(x_{r2}(g) - x_{r3}(g)) + F(x_{r4}(g) - x_{r5}(g)) \tag{3}$$

2. *DE/current-to-best/1*

$$v_i(g) = x_i(g) + F(x_{best}(g) - x_i(g)) + F(x_{r1}(g) - x_{r2}(g)) \tag{4}$$

3. *DE/best/1*

$$v_i(g) = x_{best}(g) + F(x_{r2}(g) - x_{r3}(g)) \tag{5}$$

**2.3 Correction operation**

After mutation, some components of the trial vector may go to the predefined boundary constraints. There are many solutions to tackle this problem, such as penalty schemes, resetting schemes. Here, a simple but the most frequent correction strategy (Wang et al. 2011) is used.

$$v_i(g) = \begin{cases} \min \{ x_{j,i}^{Up}, 2x_{j,i}^{Low} - v_i(g) \}, & \text{if } v_i(g) < x_{j,i}^{Low} \\ \max \{ x_{j,i}^{Low}, 2x_{j,i}^{Up} - v_i(g) \}, & \text{if } v_i(g) > x_{j,i}^{Up} \end{cases} \tag{6}$$

**2.4 Crossover**

In order to enhance the potential diversity of the population, the crossover operation happens after generating the donor vector through mutation. Crossover operators from DE exchange donor vector  $v_{j,i}(g)$ 's components with the target vector  $x_{j,i}(g)$ , which is different from the other EA that exchanges several target vectors each other from parent generation. The scheme may be described as formula (7).

$$u_{j,i}(g) = \begin{cases} v_{j,i}(g), & \text{if } rand(0, 1) \leq CR \text{ or } j = j_{rand} \\ x_{j,i}(g), & \text{otherwise} \end{cases} \tag{7}$$

where CR is the crossover rate,  $CR \in [0, 1]$ . A large CR speeds up convergence.  $j_{rand}$  is a randomly chosen integrand  $j_{rand} \in [1, D]$ , which ensures  $u_{j,i}(g)$  gets at least one component from  $v_{j,i}(g)$ .

The binomial crossover operator copies the  $j$ th parameter of the donor vector  $v_{j,i}(g)$  to the corresponding element in the trial vector  $u_{j,i}(g)$  if  $rand(0, 1) \leq CR$  or  $j = j_{rand}$ . Otherwise, it is copied from the corresponding target vector  $x_{j,i}(g)$ . The possible trial vectors due to binomial crossover are illustrated in Fig. 2. It is clear that there are 6 genes on the chromosome and  $j_{rand} = 3$ .

**2.5 Selection operation**

After above operation, a selection operation is performed to determine whether the objective function value of each target vector  $f(x_i(g))$  or the corresponding trial vector  $f(u_i(g))$  survives to the next generation, i.e., at  $g = g + 1$ . The selection operation is described as formula (8).

$$x_i(g + 1) = \begin{cases} u_i(g), & \text{if } f(u_i(g)) \leq f(x_i(g)) \\ x_i(g), & \text{otherwise} \end{cases} \tag{8}$$

**2.6 Pseudocode for the standard DE algorithm**

The detailed pseudocode of this variant is presented as follows in Table 1.

**3 The related research**

DE, nevertheless, also has the shortcomings of all other intelligent techniques. The performance of DE deteriorates with the growth of the dimensionality of the search space. DE suffers from the problem of premature convergence, where the search process may be trapped in local optima in multimodal objective function and lose its diversity. It also suffers from the stagnation problem, where the search process may occasionally stop proceeding toward the global optimum even though the population has not converged to a local optimum or any other point. Sporadically, even new individuals may enter the population, but the algorithm does not progress by

**Table 1** Pseudocode for the standard DE algorithm

---

```

01: Begin
02:   Set the generation number  $g=0$ 
03:   Create a random initial population  $x_{j,i}(g) \quad \forall i, i=1, \dots, NP \quad \forall j, j=1, \dots, D$ , each individual
04:   uniformly distributed in the range  $[x_{j,i}^{low}, x_{j,i}^{up}]$ 
05:   Evaluate  $f(x_{j,i}(g))$ 
06:   WHILE the stopping criterion is not satisfied
07:     For  $i=1$  to  $NP$ 
08:       /* Mutation step; generate a donor vector for each target vector */
09:       Select randomly  $r1 \neq r2 \neq r3$ 
10:        $v_i(g) = x_{r1}(g) + F(x_{r2}(g) - x_{r3}(g))$ 
11:       /* Correction operation */
12:       Correction()
13:       /* Crossover step; generate a trial vector for each target vector */
14:       /* Binomial crossover */
15:        $j_{rand} = \text{randint}(1, D)$ 
16:       For  $j=1$  to  $D$ 
17:         If  $\text{rand}(0,1) < CR$  or  $j = j_{rand}$ 
18:            $u_{j,i}(g) = v_{j,i}(g)$ 
19:         Else
20:            $u_{j,i}(g) = x_{j,i}(g)$ 
21:         End If
22:       End For
23:       /* Selection step */
24:       If  $f(x_i(g)) \leq f(u_i(g))$ 
25:          $x_i(g+1) = x_i(g)$ 
26:       Else
27:          $x_i(g+1) = u_i(g)$ 
28:       End If
29:     End For
30:      $g = g+1$ 
31:   End WHILE
32: End

```

---

finding any better solutions. While the global exploration ability of DE is considered adequate, its local exploitation ability is regarded weak and its convergence velocity is too low. DE is sensitive to the choice of the control parameters, and it is difficult to adjust them for different problems.

For the above drawbacks, many researchers have done many attempts to significantly enhance the overall performance of the standard DE algorithm. Yang et al. (2013) proposed a method of population adaptation which is incorporated into the jDE algorithm (Brest et al. 2009). The proposed method measured the Euclidean distances between individuals of a population which can identify the moment when the population diversity is poor or the population stagnates. Once the moment is identified, the population will be regenerated to increase diversity or to remove the stagnation issue. An improved DE named HEDE was applied to solve waveform inversion problem in Gao et al. (2014). In HEDE, a new population evolution strategy (PES) to decrease the population size based on the differences among individuals during an evolution process is embedded into the cooperative coevolutionary DE (CCDE) and obtained a new highly efficient DE. A new control parameter called cognitive learning factor (CLF) is introduced in Sharma et al. (2015). In this algorithm, a weight factor (CLF) is associated with the individual's experience in the mutation operation. In addition, the range of scale factor F is also dynamically varied

in the prescribed range. By varying CLF and F, exploration and exploitation capabilities of DE may be balanced, thus improving the performance of DE. Qinqin Fan and Xuefeng Yana proposed self-adaptive DE algorithm with discrete mutation control parameters (DMPSADE) in Fan and Yan (2015). In DMPSADE, each variable of each individual has its own mutation control parameter, and each individual has its own crossover control parameter and mutation strategy. Poikolainen et al. (2015) proposed a procedure to perform an intelligent initialization for population-based algorithms. Iipo Poikolainen and Ferrante Neri introduced a component for preprocessing the initial solutions to generate an initial population for DE algorithms. The proposed component is not dependent on any assumption on the optimization problem (except it being continuous). YuChen and WeichengXie (Ali et al. 2013) proposed a binary learning differential evolution (BLDE) algorithm which can perform well on continuous optimization problems (COOPS). The algorithm can efficiently locate the global optimal solutions by learning the information of individuals, the best explored solution and the difference between individuals in the last population. Yi et al. (2015) proposed a new DE algorithm with a hybrid mutation operator to categorize the population into two parts to process different types of mutation operators and self-adapting control parameters which are used in jDE (Brest et al. 2009) and applied in the proposed HSDE (Liu et al. 2015b) to initialize

the new population when a change is detected, and proposed a modified prediction model which uses the historical optimal sets obtained in the last two times. Guo et al. (2015) adopted a distributed topology to construct a trial vector generation pool and proposed the membership cloud model in the selection process. Multiobjective optimization algorithms have a wide range of applications, and the experts and scholars of various countries have given a wide range of attention (Giagkiozis et al. 2015; Yuan et al. 2014). At present, differential evolution algorithm has also been applied to many fields. The CPDE (differential evolutionary algorithm based on cloud population) proposed concepts of cloud group, and distribution chain concept increased the population diversity. In the algorithm, the intrusion operator will implant winners' distribution to other individuals in the process of evolution, which improves individual diversity. The cooperation mechanism and differential operation are introduced by the cooperative operator. The results prove that CPDE is an effective high-dimensional evolutionary algorithm, which has a certain advantage in optimizing the network security situation prediction model (Hu and Qiao 2016). In addition to coefficient of variation scale and cross probability strategy to improve algorithm convergence speed and solution precision, various mutation strategies are employed to increase multiplicity of the population and avoid falling to local optimal. And the results in the three-tank system show that it is effective and available, with a good feature for application in industry (Liu et al. 2015c).

A new differential evolution algorithm, APDDE, has been presented to improve optimization performance by implementing a new mutation strategy and automatically updating control parameters, which demonstrates a significant performance improvement over the JADE (Zhang and Sanderson 2009), MDE (Ali et al. 2013), CODE (Wang et al. 2011), MDESELF (Li and Yin 2014) and EPSDE (Mallipeddi et al. 2011). In order to better understand these advancements, the main contribution of JADE (Zhang and Sanderson 2009), MDE (Ali et al. 2013), CODE (Wang et al. 2011), MDESELF (Li and Yin 2014) and EPSDE (Mallipeddi et al. 2011) should be briefly summarized.

### 3.1 Adaptive DE with optional external archive (JADE)

There are three different improvements in JADE variant of adaptive differential evolution (Zhang and Sanderson 2009) extending the standardized DE concept: *current-to-pbest* mutation, archive and a new adaptive control of parameters  $F$  and CR.

#### 1. *Current-to-pbest* mutation

In JADE, the following manner will generate the mutant vector  $u_i$

$$u_i = x_i + F(x_{pbest} - x_i) + F(x_{r1} - x_{r2}) \tag{9}$$

where  $x_{pbest}$  is randomly chosen as one of the top 100  $p\%$  individuals in the current population with  $p \in (0, 1]$ , here the input parameter  $p$  is initialized to be 0.5 recommended in Zhang and Sanderson (2009). The vector  $x_{r1}$  is randomly selected from  $P(r1 \neq i)$ , and  $x_{r2}$  is randomly selected from the union  $P \cup A(r2 \neq i \neq r1)$  of the current population  $P$  and the archive  $A$ .

#### 2. Archive

The archive provides information about the progress direction and is also capable of increasing the diversity of the population. In every generation, parent individuals replaced by better offspring individuals are put into the archive; the archive is initiated to be empty, and size is controlled within NP individuals by randomly dropping surplus individuals.

#### 3. A new adaptive control of parameters $F$ and CR

$F$  and CR are the mutation factor and the crossover probability that are associated with  $x_i$  and are independently regenerated at each generation by the adaptation process; CR is generated from the normal distribution of mean  $u_{CR}$  and standard deviation 0.1, truncated to  $[0, 1]$ .  $F$  is generated from a Cauchy distribution with location parameter  $u_F$  and scale parameter 0.1 and then truncated to be 1 if  $F \geq 1$  or regenerated if  $F \leq 0$ .

### 3.2 Improving DE algorithm by synergizing different improvement mechanisms (MDE)

Ali et al. (2013) proposed a simple and modified DE framework, called MDE, which is a fusion of three modifications in DE: opposition-based learning (OBL), tournament method for mutation and single population structure.

#### 1. Opposition-based learning (OBL) for generating initial population

OBL helps in giving a good initial start to DE; if  $X = (x_1, x_2, \dots, x_n)$  is a point in  $n$ -dimensional space and  $x_i \in [l_i, u_i] \forall i \in \{1, 2, \dots, n\}$ , then the opposite point  $X' = (x'_1, x'_2, \dots, x'_n)$  is completely defined by its components

$$x'_i = l_i + u_i - x_i \tag{10}$$

Here  $l_i$  and  $u_i$  indicate the lower and upper bounds of the variables. Then, evaluate the fitness of both points  $f(X)$  and

$f(X')$ . If  $f(X') \leq f(X)$ , then replace  $X$  with  $X'$ ; otherwise, continue with  $X$ .

## 2. Tournament method for mutation

A modified mutation equation is reformulated and expressed as:

$$v_i(g) = x_{tb}(g) + F \cdot (x_{r2}(g) - x_{r3}(g)). \quad (11)$$

where  $x_{tb}(g)$  is the point with the best fitness function value among the three distinct points  $x_{r1}$ ,  $x_{r2}$ , and  $x_{r3}$  which are selected randomly from the population corresponding to target point  $x_i$ . Here assume that  $x_{tb}(g) = x_{r1}$ .

## 3. Single population structure

In the single structure of DE, only one population is maintained and the individuals are updated as the newly found better solutions can take part in the mutation and crossover operation in the current generation itself as opposed to basic DE (where two populations are maintained and the better solutions take part in mutation and crossover operations in the next generation).

### 3.3 DE with composite trial vector generation strategies and control parameters (CODE)

CODE is a simple but efficient algorithm (Wang et al. 2011). Yong Wang only used two candidate pools: the strategy candidate pool and the parameter candidate pool. The strategy candidate pool contains three trial vector generation strategies (*rand/1/bin*, *rand/2/bin*, *current-to-rand/1*), and three control parameters ( $[F = 1.0, CR = 0.1]$ ,  $[F = 1.0, CR = 0.9]$ ,  $[F = 0.8, CR = 0.2]$ ) constitute the parameter candidate pool. Then, at each generation, each target vector uses the three trial vector generation strategies; each combines with a control parameter setting randomly chosen from the parameter candidate pool, to generate three trial vectors, and the chosen best one enters the next generation if it is better than its target vector.

### 3.4 Modified DE with self-adaptive parameters method (MDESELF)

The modified algorithm in Li and Yin (2014) named MDE is the same as in Ali et al. (2013); so in order to be distinguished, we shall refer to it as MDESELF in the present study. Overall, there are also three modifications are proposed in Li and Yin (2014): modified mutation strategy, randomized scale factor and self-adaptive crossover rate.

#### 1. Modified mutation strategy

The new mutation strategy is similar to the formula (9); the difference is that it does not use the external archive, and the random vectors are used instead of the current individual; in other words, the vectors  $x_{r2}$  in (9) are no longer randomly selected from the union  $P \cup A$ , but only from  $P$ , the current

individual  $x_i$  becomes the random vectors. In addition, the new mutation strategy is combined with the basic mutation strategy ‘rand/2/bin’(3) through some probability rules. It can be expressed as follows:

$$\begin{aligned} \text{If } \text{rand} \leq \omega \text{ then } v_i(g) &= x_{r1}(g) + F \left( x_{\text{best}}^P(g) - x_{r1}(g) \right) \\ &+ F(x_{r2}(g) - x_{r3}(g)) \\ \text{Else } v_i(g) &= x_{r1}(g) + \text{rand}(x_{r2}(g) - x_{r3}(g)) \\ &+ F(x_{r4}(g) - x_{r5}(g)) \\ \text{End if} & \end{aligned} \quad (12)$$

Here the input parameter  $\omega = g/gmax$  is the probability value and recommended in Li and Yin (2014), and  $gmax$  is the maximum generation number.

#### 2. Randomized scale factor

The value of  $F$  is generated as two Gaussian distributions:  $F1 = \text{Gaussian}(0.3, 0.3)$ ,  $F2 = \text{Gaussian}(0.7, 0.3)$ . If  $\text{rand}() \leq \text{rand}()$ , then  $F = F1$ ; otherwise,  $F = F2$ .

#### 3. Self-adaptive crossover rate

The author proposed a relative success ratio selecting one of two new parameters in the previous periods; according to it, the algorithm selects the evolution method for each individual. (Details of CR adaptation are shown in Li and Yin (2014)).

### 3.5 DE algorithm with ensemble of parameters and mutation strategies (EPSDE)

EPSDE (Mallipeddi et al. 2011) also have two candidate pools (a pool of mutation strategies and a pool of control parameters), which is similar to CODE, but the strategy candidate pool includes *DE/best/2/bin*, *DE/rand/1/bin*, *DE/current-to-rand/1/bin* and parameter candidate pool in which CR values range from 0.1–0.9 in steps of 0.1; the  $F$  values range from 0.4 to 0.9 in steps of 0.1, which are different from CODE. Each member (target vector) is randomly assigned a mutation strategy and united parameter values taken from the respective pools to generate offspring (trial vector). If  $f(x_i(g))$  (function values of target vectors)  $\leq f(u_i(g))$  (function values of target vectors), then the combination of the mutation strategy and the parameter values is stored in the successful combinations pool; otherwise, reinitialize the mutation formula from the respective pools or from the successful combinations pool with equal probability. In Fig. 3, we will see the process which generates the successful combination pool.

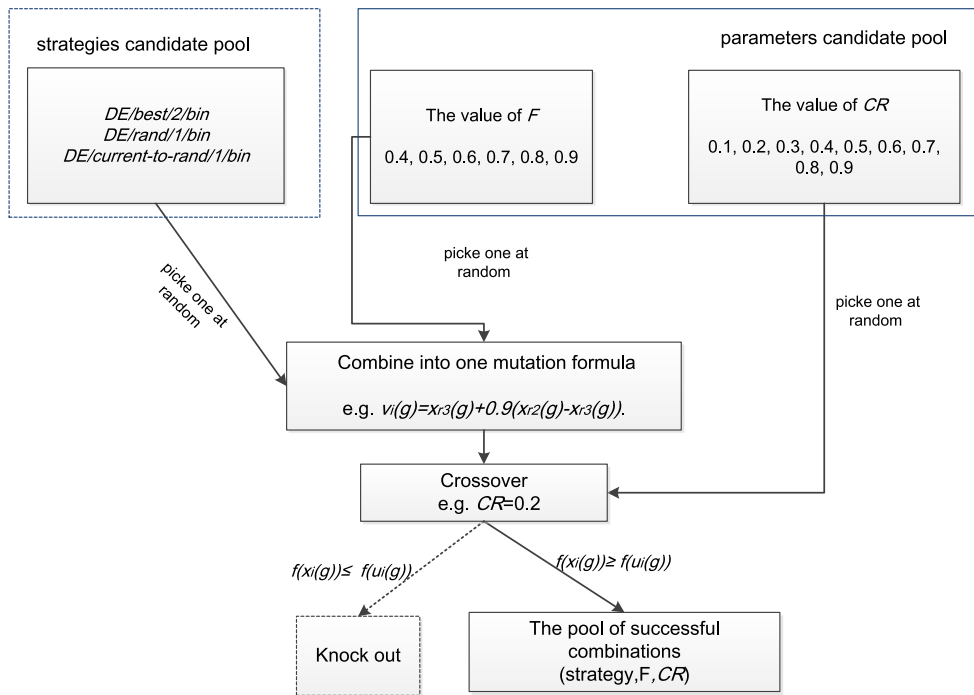


Fig. 3 Process of generating the successful combination pool

### 4 The APDDE

This section proposes APDDE (a self-adaptive differential evolution algorithm).

The APDDE imitates the sniffing behavior of canine animals like, especially, dogs in the biological world. We know dogs have a unique way of sniffing around when they walk or hunt for food; inspired by this, we apply this behavior to differential evolution algorithm, so the whole algorithm optimization process has certain direction. In addition, we propose a new mutation strategy *DE/local-to-best/1* and control *F* and *CR* in an adaptive manner for each individual to balance between the exploration and the exploitation. In general, three modifications are proposed in this paper and they are elucidated as follows.

#### 4.1 DE/Current-to-lbest/1

The *DE/Current-to-lbest/1* strategy is a variant of the *DE/Current-to-pbest* strategy. In the *DE/Current-to-pbest* strategy, in order to enrich the individuals of the population, the individuals are mutated under the guidance of one individual which is randomly chosen from the top 100*p*% individuals in the current population with  $p \in (0, 1]$ . However, in *DE/Current-to-lbest/1*, the population is attracted by the multiple locally best individuals instead of one of the top 100*p*% best individuals.

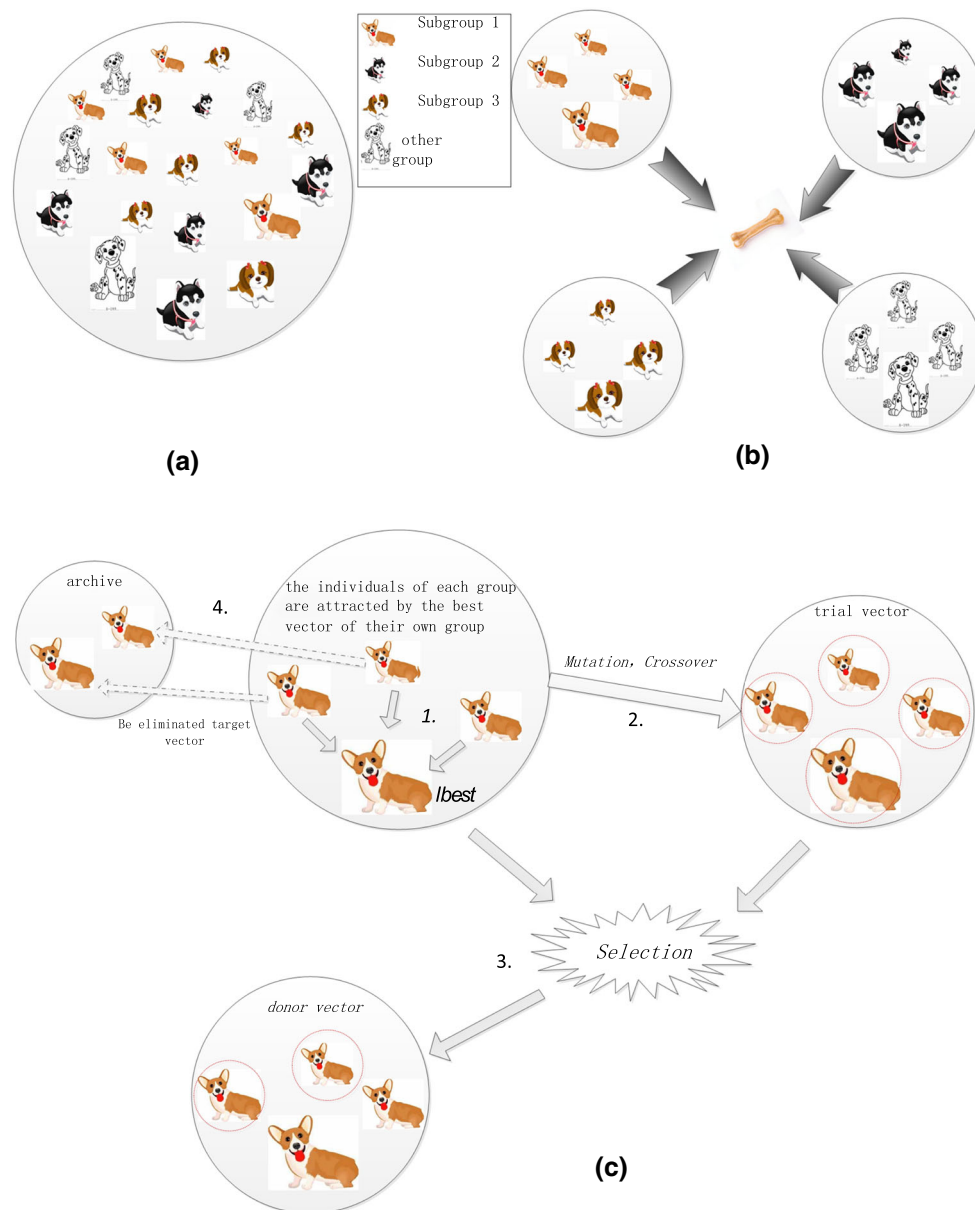
In the proposed strategy, combining the idea of SFLA, we make improvements on aspects of the population recon-

struction. The whole population is divided into a predefined number of groups simply based on the vector indices. Each group contains the same number of vectors, and the members of each group are kept unchanged during the algorithm’s iteration. The members of each equal-size groups, in fact, are assigned randomly from the population, since the vector indices are sorted randomly during initialization. The population is divided into several subgroups; in iterative process, each subgroup can obtain a local optimal value, and then, the local information obtained by subgroups is shared among the population in order to increase diversity as much as possible to make the population to avoid falling into local optimum and also, at the same time, to obtain the global optimal.

All individuals within the population are marked with random numbers, each group with 10 individuals, and each individual is assigned to different groups according to their random numbers. For example, the individuals with 1–10 numbers are assigned to the first group, 10–20 are assigned to the second group, and so on. The characteristics of each group are random, and no better or worse, because the groups are not divided by the sorting fitness value. The reason for this randomization is that it increases not only the diversity of the entire population, but also diversity of each packet. And the *DE/Current-to-lbest/1* strategy can be expressed as follows:

$$u_i = x_i + F_i(x_{lbest} - x_i) + F_i(x_{r1} - x_{r2}), \tag{13}$$

where the subscript *lbest* denotes the best vector in the group with respect to target vector *i*,  $x_{r1}$  is selected from the current



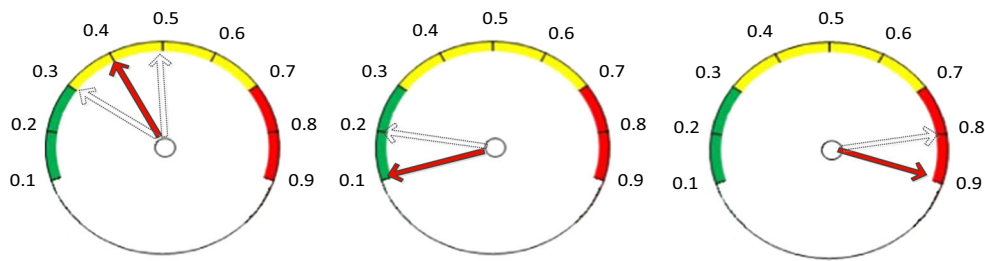
**Fig. 4** Grouping and procedures of each subgroup. **a** Random initial. **b** Dividing the population into several subgroup. **c** Exploration in each subgroup

population, while  $x_{r,2}$  is randomly chosen from the union, the current population and archive (initialed to be empty, the parent solutions that fail in the selection process are added to the archive, and the archive size are kept at NP). The vivid description of grouping and procedures of each subgroup is shown in Fig. 4. Among the serial numbers, 1, 2, 3 and 4 is the order of exploration of each group. In this paper, this new mutation strategy is integrated into the basic mutation strategy *DE/rand/1/bin* to form a strategy pool.

## 4.2 Parameter adaptation

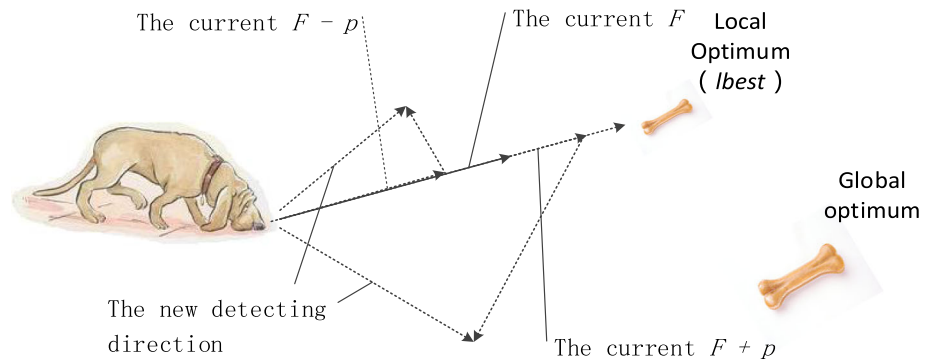
Storn and Price suggested that  $F$  should be 0.6 or 0.5 as a good initial choice and the value of  $F$  smaller than 0.4 or larger than 1.0 will lead to decrease in convergence speed (Storn and Price 1997), In contrast, Ronkkonen et al. (2005) said that  $F = 0.9$  would be a good initial choice and  $F$  should be chosen from the range [0.4, 0.95]. Storn and Price argued that the value of CR should be set to 0.1 or 0.9 and thought 1.0 can be tried to increase the convergence speed,





**Fig. 5** Generating the  $CR_{de}$  and  $CR_{ad}$  when  $p = 0.1$

**Fig. 6** Process of detecting in algorithm



while in Ronkkonen et al. (2005), the value of CR choice from the range (0.3, 0.9) will be best. CR should be between 0.0 and 0.2 for separable functions, while for multimodal and non-separable, the best choice is from the range (0.9, 1.0).

In general, in our APDDE algorithm, the parameter  $F$  will be chosen in the range [0.5, 1] in the steps of  $p$  (in this paper, the value of  $p$  is 0.1), and the pool of CR values taken in the range [0.4, 0.9] in steps of  $p$  is similar to EPSDE (Mallipeddi et al. 2011). In the algorithm, each vector in the population is extended with parameter  $F$ , CR (i.g.  $x_{1,i}(g), x_{2,i}(g), \dots, x_{D,i}(g), F_{i,g}, CR_{i,g}$ ) and the value is randomly selected from the respective pool in the initialization. During evolution, the self-adaptive process of  $F$  and CR is as follows:

1.  $F_{de} = F_{i,g} - p,$   
 $F_{ad} = F_{i,g} + p;$   
 If  $F_{i,g} = 0.5$   
 then  $F_{de} = 0.5.$   
 If  $F_i = 1$   
 then  $F_{ad} = 1.$
2.  $CR_{de} = CR_{i,g} - p,$   
 $CR_{ad} = CR_{i,g} + p;$   
 If  $CR_{i,g} = 0.1$   
 then  $CR_{de} = 0.1.$   
 If  $CR_{i,g} = 0.9$   
 then  $CR_{de} = 0.9.$

In Fig 5, the process of generating the  $CR_{de}$  and  $CR_{ad}$  is shown. The red indicator hand is the current parameter  $CR_{i,g}$ .

In each iteration, the current parameters  $F_{i,g}$  and  $CR_{i,g}$  will generate, respectively, the corresponding detection values:  $F_{de}$  and  $F_{ad}$ ,  $CR_{de}$  and  $CR_{ad}$ . The vivid description of the process is show in Figs. 6 and 7, respectively.

The dashed curves display the contours of the optimization problem.  $V_{i,1}$  and  $V_{i,2}$  are the mutation vector generated for individual  $X_i$  by the associated mutation factor  $F_{de}$  and  $F_{ad}$ .

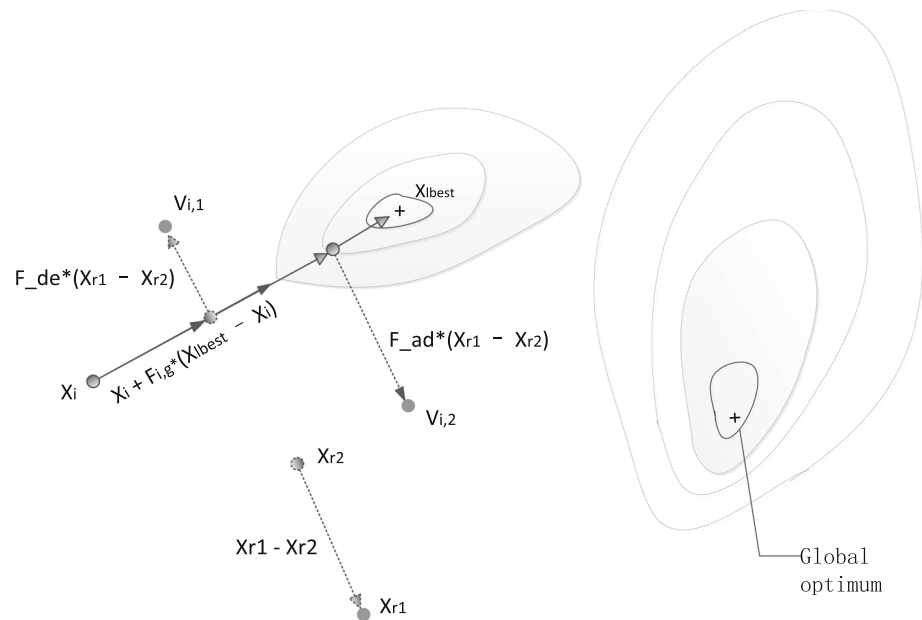
### 4.3 Procedure of APDDE

According to 4.2, each individual in the initial population will generate four groups of control parameters:  $[F_{de}, CR_{de}]$ ,  $[F_{de}, CR_{ad}]$ ,  $[F_{ad}, CR_{de}]$  and  $[F_{ad}, CR_{ad}]$ . Each group is integrated into the mutation strategy in the strategy pool and generates eight groups:

- $[DE/Current - to - lbest/1, F_{de}, CR_{de}]$ ,  $[DE/Current - to - lbest/1, F_{de}, CR_{ad}]$ .
- $[DE/Current - to - lbest/1, F_{ad}, CR_{de}]$ ,  $[DE/Current - to - lbest/1, F_{ad}, CR_{ad}]$ .
- $[DE/rand/1, F_{de}, CR_{de}]$ ,  $[DE/rand/1, F_{de}, CR_{ad}]$ .
- $[DE/rand/1, F_{ad}, CR_{de}]$ ,  $[DE/rand/1, F_{ad}, CR_{ad}]$ .

The population individuals (target vectors) produce offspring (trial vectors) using these eight assigned combinations. If the best of these eight generated trial vectors produced is better than the target vector, the corresponding parameter values are retained with trial vector which

**Fig. 7** Illustration of the *DE/current-to-lbest/l* mutation strategy adopted in APDDE



becomes the parent (target vector) in the next generation. In APDDE, new mutation strategies *DE/Current-to-lbest/l* and basic mutation strategy *DE/rand/l/bin* are applied, and mutation operator  $F$ , crossover  $CR$ , produce their different detection values  $F_{de}$ ,  $F_{ad}$ ,  $CR_{de}$  and  $CR_{ad}$ ; 8 kinds of strategies pool are produced through Cartesian operation. According to above chart of the APDDE, the pseudocode of the APDDE is shown in Table 2. In the iterative process, each variation vector carries out crossover operation to form eight intermediate vector individuals, and the fitness value of each new produced intermediate vector individuals is obtained with a variable detection. Choose the best fitness of intermediate individual from eight variables. The more different mutation strategies with different intermediate vectors (within certain limits), the greater possibility of reaching the optimal direction. At the same time, different variations can also maintain the diversity of population.

## 5 Experimentation

In this section, the proposed APDDE is compared with the EPSDE, JADE, MDE, CODE, MDESELF.

### 5.1 Description of benchmark functions

Those 13 test instances were proposed in the CEC 2015 Competition on Learning-based Real-Parameter Single Objective Optimization, and 7 well-known basic benchmarks have been selected to verify the performance of the APDDE.

1.  $f_1(x) = \sum_{i=1}^N x_i^2$  is continuous, convex and symmetric function.
2.  $f_2(x) = \sum_{i=1}^N (x_i^2 - 10 \cos(2\pi x_i) + 10)$ , Rastrigin function has several local minima. It is highly multi-

modal, but locations of the minima are regularly distributed.

3.  $f_3(x) = -\sum_{i=1}^N \sin(x_i) [\sin(ix_i^2/\pi)]^{2m}$ , Michalewicz function has dimensional (here is 50) local minima. The parameter  $m$  defines the steepness of the valleys and ridges; a larger  $m$  leads to a more difficult search. The  $m$  is set to 10 in this paper.
4.  $f_4(x) = \sum_{i=1}^N |x_i| + \prod_{i=1}^n |x_i|$ , Schwefel's problem 2.22 function is a unimodal problem whose global minimum is 0.
5.  $f_5(x) = \left( \sum_{i=1}^5 i \cos((i+1)x_1 + i) \right) \left( \sum_{i=1}^5 i \cos((i+1)x_2 + i) \right)$ , it is a low-dimensional multimodal function with a few local optima.
6.  $f_6(x) = -20 \exp\left(-0.2 \sqrt{\frac{1}{n} \sum_{i=0}^N x_i^2}\right) - \exp\left(\frac{1}{n} \cos(2\pi x_i)\right) + 20 + e$ , Ackley function has one narrow global optimum basin and many minor local optima.
7.  $f_7(x) = \sum_{i=1}^N \sum_{j=1}^i x_j^2$  is continuous, convex and unimodal. It is an extension of the axis parallel hyper-ellipsoid function, also referred as the sum squares function.
8.  $f_8(x) = \sum_{i=1}^N (10^6)^{i-1/n-1} z_i^2 + 100$ ,  $z = M_1(x - o_1)$ ,  $o_1 = [o_{11}, o_{12}, \dots, o_{1n}]$ : the shifted global optimum. Its property is non-separable, quadratic ill-conditioned.
9.  $f_9(x) = z_1^2 + 10^6 \sum_{i=2}^N z_i^2 + 200$ ,  $z = M_2(x - o_2)$ ,  $o_2 = [o_{21}, o_{22}, \dots, o_{2n}]$ : the shifted global optimum. Its property is non-separable, smooth but narrow ridge.
10.  $f_{10}(x) = -20 \exp\left(-0.2 \sqrt{\frac{1}{n} \sum_{i=0}^N z_i^2}\right) - \exp\left(\frac{1}{n} \cos(2\pi z_i)\right) + 20 + e + 300$ ,  $z = M_3(x - o_3)$ ,  $o_3 = [o_{31}, o_{32}, \dots, o_{3n}]$ : Its property is shifted global optimum and non-separable.
11.  $f_{11}(x) = \sum_{i=1}^N (z_i^2 - 10 \cos(2\pi z_i) + 10) + 400$ ,  $z = M_4(5.12(x - o_4)/100)$ ,  $o_4 = [o_{41}, o_{42}, \dots, o_{4n}]$ : the

**Table 2** Pseudocode for the APDDE algorithm

```

01: Begin
02:   Set the generation number  $g=0$ 
03:   Create a random initial population  $x_{j,i}(g) \forall i, i=1, \dots, NP \forall j, j=1, \dots, D$ , each individual
      uniformly distributed in the range  $[x_{j,i}^{low}, x_{j,i}^{up}]$ 
04:    $F\_Pool=[0.5,0.6,0.7,0.8,0.9,1]$ 
05:    $CR\_Pool=[0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9]$ 
06:    $A=\emptyset$ 
07:   Each population individuals is assigned with the associated parameter values are chosen
      randomly from the corresponding pool  $F\_Pool, CR\_Pool$ 
08:   Divide the population into  $T$  num subpopulation
09:   WHILE the stopping criterion is not satisfied
10:     find the  $X_{lbest,t}$  in each subpopulation
11:     For  $i=1$  to  $NP$ 
          /* Mutation operation; */
12:       Select randomly  $r1 \neq r3 \neq r4 \neq r5 \neq i$  from current population  $P$ 
13:       Select randomly  $r2(r2 \neq r3 \neq r4 \neq r5 \neq i)$  from  $P \cup A$ 
14:       According to the current parameter  $F_i$  and  $CR_i$ , generate the values of sniffer:  $F\_de$  and
           $F\_ad, CR\_de$  and  $CR\_ad$ , integrating into the mutation strategy in the strategy pool and
          produce four mutated vectors
15:        $v_{i,1}(g) = x_i(g) + F\_de(X_{lbest,t} - x_i(g)) + F\_de(x_{r1}(g) - x_{r2}(g))$ 
16:        $v_{i,2}(g) = x_{r3}(g) + F\_de(x_{r4}(g) - x_{r5}(g))$ 
17:        $v_{i,3}(g) = x_i(g) + F\_ad(X_{lbest,t} - x_i(g)) + F\_ad(x_{r1}(g) - x_{r2}(g))$ 
18:        $v_{i,4}(g) = x_{r3}(g) + F\_ad(x_{r4}(g) - x_{r5}(g))$ 
19:       Generate the current mutated vectors
20:        $v_{i,current1}(g) = x_i(g) + F_i(X_{lbest,t} - x_i(g)) + F_i(x_{r1}(g) - x_{r2}(g))$ 
21:        $v_{i,current2}(g) = x_{r3}(g) + F_i(x_{r4}(g) - x_{r5}(g))$ 
22:       Correction() /* Correction operation */
23:       Crossover() /* Crossover operation */
          Generate eight trial vector  $u_{i,1}(g), u_{i,2}(g), u_{i,3}(g), u_{i,4}(g), u_{i,5}(g), u_{i,6}(g), u_{i,7}(g), u_{i,8}(g)$ 
          and two current trial vector  $u_{i,current1}(g), u_{i,current2}(g)$ 
          /* Selection operation */
24:       Find the best trial vector:  $u_{best}(g) = \text{sort}(f(u_{i,1}(g)), f(u_{i,2}(g)), f(u_{i,3}(g)), f(u_{i,4}(g)),$ 
           $f(u_{i,5}(g)), f(u_{i,6}(g)), f(u_{i,7}(g)), f(u_{i,8}(g)), f(u_{i,current1}(g)), f(u_{i,current2}(g)))$ 
25:       If  $f(x_i(g)) \leq f(u_{best}(g))$ 
26:          $x_i(g+1) = x_i(g)$ 
27:       Else
28:          $x_i(g+1) = u_{best}(g); x_{i,g} \rightarrow A$ ; the corresponding parameter value  $\rightarrow F_i, CR_i$ 
29:       End If
30:     End For
31:      $g = g+1$ 
32:   End WHILE
33: End

```

$T$  the number of subpopulation  
 $A$  the archive  
 $X_{lbest,t}$  the best individual in  $t$  subpopulation,  $0 < t < T$   
 $sort()$  find the best trial vector basis on benchmark function

shifted global optimum. Its property is non-separable, and local optima's number is huge. 12.  $f_{12}(x) = 418.9829 \times n - \sum_{i=1}^N g(y_i) + 500, y_i = z_i + 4.209687462275036e + 002,$

$$g(y_i) = \begin{cases} y_i \sin(|y_i|^{1/2}) & \text{if } |y_i| \leq 500 \\ ((500 - \text{mod}(y_i, 500)) \sin(\sqrt{|500 - \text{mod}(y_i, 500)|})) - \frac{(y_i - 500)^2}{10000n} & \text{if } y_i > 500 \\ ((\text{mod}(|y_i|, 500) - 500) \sin(\sqrt{|\text{mod}(|y_i|, 500) - 500|})) - \frac{(y_i + 500)^2}{10000n} & \text{if } y_i < -500 \end{cases},$$

**Table 3** Dimensions, search ranges, and brief descriptions of test functions

Func.#	Brief descriptions of test benchmark function	Uni./multi	Dimension	Range	Optimums.
$f_1(x)$	Sphere function	Unimodal	50	$[-100, 100]^n$	0
$f_2(x)$	Rastrigin function	Multimodal	50	$[-10, 10]$	0
$f_3(x)$	Michalewicz function	Multimodal	50	$[0, \pi]$	-9.66
$f_4(x)$	Schwefel's problem 2.22 function	Unimodal	50	$[-10, 10]$	0
$f_5(x)$	Shub ert function	Multimodal	2	$[-10, 10]$	-186.73
$f_6(x)$	Ackle function	Multimodal	50	$[-600, 600]$	0
$f_7(x)$	Rotated hyper-ellipsoid function	Unimodal	50	$[-65.536, 65.536]$	0
$f_8(x)$	F1 (Rotated high conditioned elliptic function)	Unimodal	50	$[-100, 100]$	100
$f_9(x)$	F2 (Rotated cigar function)	Unimodal	50	$[-100, 100]$	200
$f_{10}(x)$	F3 (Shifted and rotated Ackley's function)	Multimodal	50	$[-100, 100]$	300
$f_{11}(x)$	F4 (Shifted and rotated Rastrigin's function)	Multimodal	50	$[-100, 100]$	400
$f_{12}(x)$	F5 (Shifted and rotated Schwefel's function)	Multimodal	50	$[-100, 100]$	500
$f_{13}(x)$	F6 (Hybrid function 1 ( $N = 3$ ))	Multimodal	50	$[-100, 100]$	600
$f_{14}(x)$	F7 (Hybrid function 2 ( $N = 4$ ))	Multimodal	50	$[-100, 100]$	700
$f_{15}(x)$	F8 (Hybrid function 3 ( $N = 5$ ))	Multimodal	50	$[-100, 100]$	800
$f_{16}(x)$	F9 (Composition function 1 ( $N = 3$ ))	Multimodal	50	$[-100, 100]$	900
$f_{17}(x)$	F10 (Composition function 2 ( $N = 3$ ))	Multimodal	50	$[-100, 100]$	1000
$f_{18}(x)$	F13 (Composition function 5 ( $N = 5$ ))	Multimodal	50	$[-100, 100]$	1300
$f_{19}(x)$	F14 (Composition function 6 ( $N = 7$ ))	Multimodal	50	$[-100, 100]$	1400
$f_{20}(x)$	F15 (Composition function 7 ( $N = 10$ ))	Multimodal	50	$[-100, 100]$	1500

$z = M_5(1000(x - o_5)/100)$ ,  $o_3 = [o_{31}, o_{32}, \dots, o_{3n}]$ : the shifted global optimum. Its property is non-separable, local optima's number is huge, and second better local optimum is far from the global optimum.

13.  $f_{13}(x)$  is hybrid function; three basic functions used to construct it are modified Schwefel's function, Rastrigin's function and high conditioned elliptic function.
14.  $f_{14}(x)$  is hybrid function; 4 basic functions used to construct it are Griewank's function, Weierstrass function, Rosenbrock's function and Scaffer's F6 function.
15.  $f_{15}(x)$  is hybrid function; 5 basic functions used to construct it are Scaffer's F6 function, HGBat function, Rosenbrock's function, modified Schwefel's function and high conditioned elliptic function.
16.  $f_{16}(x)$ ,  $f_{17}(x)$ ,  $f_{18}(x)$ ,  $f_{19}(x)$ ,  $f_{20}(x)$  is composition function and has different properties for different variables subcomponents; the details of constructing such functions are presented in Liang et al. (2015).

All the definitions of the benchmark functions are given according to the number in Table 3.

## 5.2 Experimental setting and parameterization

In order to ensure fairness, the initial conditions of each algorithm are consistent, such as initial velocity. The population size is set to 100. The size of the subgroup in APDDE is set to 10, and the size of archive in APDDE and JADE is set to 100. The 50-dimensional test function of 25 independent experiments demonstrate their general performances.

Experimental environment configuration: operation system is Windows 7; minimum memory is 4G; processor type is Intel-Core-i5; development tool and version is MATLAB-R2012a.

If it is not easy to distinguish the effect of different functions in figure, the value of ordinate should be set into the

$\log(f(x) - Y)$ , in which  $Y$  is a constant. We also can set  $Y = f(x^*)$ ; then,  $Y$  or  $f(x^*)$  represents the theoretical optimal value. Of course, if the result in the figure is easy to distinguish, the value of ordinate is function value or fitness.

### 5.3 Computational results and discussion

For each test function, Table 4 shows the comparison results of APDDE, EPSDE, JADE, MDE, CODE and MDESELF, in which *mean* indicates the average optimum function value

**Table 4** Mean and std difference in test functions about seven algorithms

Function	Indicator	Comparison of each DE					
		APDDE	EPSDE	JADE	MDE	CODE	MDESELF
$f_1(x)$	Mean	<b>9.65E-244</b>	1.10E-39	1.31E-164	6.18E-84	7.24E-20	1.86E-51
	Std	<b>0.00E+00</b>	1.85E-39	<b>0.00E+00</b>	7.85E-84	4.90E-20	9.25E-51
$f_2(x)$	Mean	<b>0.00E+00</b>	1.11E+02	2.85E+02	2.54E+02	8.74E+01	1.42E+02
	Std	<b>0.00E+00</b>	4.51E+01	3.02E+01	6.48E+01	3.92E+00	4.67E+00
$f_3(x)$	Mean	<b>-4.96E+01</b>	-3.22E+01	-2.02E+01	-1.90E+01	-3.62E+01	-3.24E+01
	Std	<b>2.31E-03</b>	7.43E-01	1.11E+00	1.84E+00	7.31E-01	1.03E+00
$f_4(x)$	Mean	<b>3.20E-32</b>	2.02E-03	5.34E-25	6.77E-11	3.81E+00	6.17E+00
	Std	<b>2.88E-32</b>	3.93E-03	4.34E-25	6.81E-11	8.91E-01	5.76E+00
$f_5(x)$	Mean	<b>-2.09E+02</b>	-1.87E+02	-1.87E+02	-1.87E+02	-1.87E+02	-1.87E+02
	Std	3.15E+00	2.20E-14	2.20E-14	2.01E-14	<b>0.00E+00</b>	2.98E-14
$f_6(x)$	Mean	<b>2.00E+01</b>	2.01E+01	2.10E+01	2.11E+01	2.01E+01	2.03E+01
	Std	<b>2.13E-12</b>	1.29E-01	6.30E-02	3.36E-02	6.48E-03	2.59E-02
$f_7(x)$	Mean	<b>5.29E-77</b>	6.48E-10	1.51E-51	2.64E-24	5.11E-03	1.45E-14
	Std	<b>8.65E-77</b>	6.99E-10	1.71E-51	3.36E-24	1.66E-03	2.80E-14
$f_8(x)$	Mean	2.73E+06	3.82E+07	2.36E+07	8.35E+05	8.35E+05	<b>3.04E+05</b>
	Std	1.40E+06	5.90E+06	1.07E+07	4.26E+05	1.18E+07	<b>1.42E+05</b>
$f_9(x)$	Mean	<b>2.00E+02</b>	1.10E+03	2.97E+03	2.01E+02	6.03E+05	<b>2.00E+02</b>
	Std	<b>9.19E-10</b>	8.93E+02	2.65E+03	3.36E-01	1.30E+05	5.03E-06
$f_{10}(x)$	Mean	<b>3.20E+02</b>	3.21E+02	3.21E+02	3.21E+02	3.21E+02	3.21E+02
	Std	<b>1.69E-02</b>	3.55E-02	4.17E-02	4.50E-02	3.46E-02	4.02E-02
$f_{11}(x)$	Mean	<b>4.88E+02</b>	6.50E+02	7.93E+02	7.19E+02	6.46E+02	5.88E+02
	Std	1.13E+01	<b>1.10E+01</b>	2.55E+01	6.04E+01	1.31E+01	1.60E+01
$f_{12}(x)$	Mean	<b>5.87E+03</b>	9.76E+03	1.47E+04	1.35E+04	8.95E+03	8.76E+03
	Std	<b>2.22E+02</b>	3.45E+02	5.61E+02	2.77E+02	2.28E+02	4.09E+02
$f_{13}(x)$	Mean	2.18E+05	1.82E+06	7.02E+06	5.00E+04	3.98E+06	<b>4.27E+03</b>
	Std	1.46E+05	6.37E+05	2.72E+06	4.36E+04	1.11E+06	<b>2.05E+03</b>
$f_{14}(x)$	Mean	<b>7.25E+02</b>	7.43E+02	7.57E+02	7.42E+02	7.40E+02	7.31E+02
	Std	<b>1.36E+00</b>	7.59E+00	1.07E+01	1.07E+01	1.43E+01	1.57E+01
$f_{15}(x)$	Mean	6.47E+04	1.37E+04	4.12E+06	3.97E+03	8.00E+04	<b>1.74E+03</b>
	Std	5.80E+04	6.03E+03	2.04E+06	1.36E+03	2.21E+04	<b>2.73E+02</b>
$f_{16}(x)$	Mean	<b>9.05E+02</b>	1.01E+03	1.01E+03	1.01E+03	1.01E+03	1.01E+03
	Std	<b>2.29E-01</b>	2.98E-01	4.07E-01	2.70E-01	3.98E-01	2.77E-01
$f_{17}(x)$	Mean	6.68E+03	1.03E+04	9.54E+05	3.12E+03	1.25E+04	<b>2.32E+03</b>
	Std	<b>3.67E+02</b>	1.54E+03	6.70E+05	6.53E+02	1.41E+03	3.69E+02
$f_{18}(x)$	Mean	<b>1.51E+03</b>	<b>1.51E+03</b>	1.53E+03	1.52E+03	<b>1.51E+03</b>	<b>1.51E+03</b>
	Std	<b>1.56E+00</b>	3.86E+00	2.59E+00	3.21E+00	2.37E+00	3.89E+00
$f_{19}(x)$	Mean	<b>9.58E+03</b>	6.36E+04	6.50E+04	6.68E+04	6.15E+04	5.93E+04
	Std	<b>1.95E+03</b>	9.07E+03	1.18E+04	6.84E+03	3.10E+03	2.86E+03
$f_{20}(x)$	Mean	<b>1.60E+03</b>	<b>1.60E+03</b>	<b>1.60E+03</b>	<b>1.60E+03</b>	<b>1.60E+03</b>	<b>1.60E+03</b>
	Std	2.41E-13	4.51E-08	3.41E-13	<b>1.39E-13</b>	2.19E-03	7.71E-12

of the 25 runs and *std* means the standard deviation of the optimum function value.

Figure 8 illustrates the detailed convergence curves of APDDE, EPSDE, JADE, MDE, CODE and MDESELF for the 20 benchmark functions, which were drawn by using the average value of the 25 runs.

From Fig. 8 and Table 4, we can observe that the convergence speed and accuracy of APDDE are much better than other differential evolution algorithms with the same dimension in solving great majority benchmark functions.

It has  $d!$  local optima for the  $d$ th-dimensional Michalewicz function. The parameter  $d$  actually measures the ‘steepness’ of the valleys or edges. For large  $d$ , the function behaves like a needle in the haystack; the function values for points outside the narrow peaks give very little information on the location of the global optimum. Table 4 shows that EPSDE, JADE, MDE, CODE and MDESELF have trouble with finding the needle. On the contrary, the hybrid scheme MFSLA-EO is very attractive in that it is able to reach the global optimum quickly with a higher success ratio. For Schwefel’s problem 2.22, APDDE’s accuracy and stability are the highest, but the convergence rate in early stage is lower than EPSDE’s. When iteration reaches to 400 times, APDDE, JADE and MDE began to converge fast. APDDE is the best, and JADE is the second best. At last, APDDE reached the best accuracy, and on the whole, APDDE outperforms others. Another multimodal function, Shubert function, in the initial stage of algorithm, APDDE’s convergence speed is the fastest, but in 100–600 times, six algorithms also fall into the local optimum, and from 600 times, APDDE algorithm is able to find the hidden optimal value, jump out of local optimum and continue to optimize. Of course, stability is far from CODE; this is because the CODE falls into the same local optimal value each time, but not jump out of it. For Schwefel’s problem, APDDE and MDESELF have the same best value in  $f_9$ . In  $f_{10}$ , APDDE is the best from the accuracy to the convergence rate; CODE is better in handling the multiple peak function. While in  $f_{13}$ , APDDE’s performance is not good, EPSDE has the best performance, followed by MDE; before the iteration reaches 700 times, MDE is better than EPSDE, but in the later stage of the algorithm, EPSDE has outstanding performance. The experimental results of  $f_{15}$  are similar to  $f_6$ , in which their mixed function is almost the same; MDESELF shows a good advantage, followed by MDE. And APDDE performance is poor, that is to say, APDDE is weak to deal with the relatively strong mixture test functions. For  $f_{17}$ , we can see that the exact value of MDESELF is relatively good, but before reaching 800 times, the convergence rate is weaker than MDE. But the stability of APDDE outperforms others. APDDE, EPSDE, CODE and MDESELF get the same best value on  $f_{18}$ . F20 contains 10 functions, which has a very high comprehensive performance. On the one hand, the algorithm on the test functions

generally showed poor performance; this results can also reflect that various algorithms are trapped in the local optimum; on the other hand, it can also test the algorithm’s limit ability.

In multi-modal functions, APDDE provides equal or better results except  $f_8, f_{13}, f_{15}, f_{17}$ . In unimodal functions, the results indicate that APDDE is better than the other algorithms for 50 dimensions. These results indicate that in 50 dimensions, the overall performance of APDDE is better than other algorithms.

APDDE exhibits similar performance on  $f_{18}(x)$  and  $f_{20}(x)$  test functions. In other words, APDDE has stronger global search ability and faster convergence speed compared with EPSDE, JADE, MDE, CODE and MDESELF. But its performance is not perfect at  $f_8(x), f_{13}(x), f_{15}(x), f_{17}(x)$ . Clearly, MDESELF is the best among others algorithms on these functions; MDE is the second best. This may be because MDESELF uses complex self-adaptive crossover rate, while the adaptive method of APDDE mainly pursues saving time. MDE use OBL for generating initial population at each iteration, but APDDE generates initial population only once and does not change during evolution.

In order to show advantages of the APDDE, we do some  $T$  test and  $F$  test by using the data from the experiment and compare mean optimum values by  $T$  test matched-pairs mean analysis,  $F$  test matched-pairs standard difference analysis and Wilcoxon signed ranks test. Wilcoxon matched-pairs signed ranks test is a nonparametric test employed in hypothetical testing situation involving two samples (Kiranyaz et al. 2011), and it is a pair-wise test that can be used to detect significant differences between the behavior of APDDE and EPSDE, JADE, MDE, CODE and MDESELF with fixed study parameters. The data column of mean optimization value obtained by APDDE was noted as the first sample, while the results column obtained by EPSDE, JADE, MDE, CODE, MDESELF with fixed study parameter was noted as the second sample, respectively.

Let  $\theta_D$  be the median of the difference in the underlying population represented by the two samples. Let  $H_0$  be the null hypothesis,  $H_0 : \theta_D = 0$ ; which means that there is no difference between the means of two samples. The alternative hypothesis is  $H_1 : \theta_D < 0$ , which is a directional hypothesis.

We use the SPSS11.0 (Statistic Package for Social Science) software, which calculates the  $T$  test and  $F$  test, as shown in Tables 5 and 6.

We reject the null hypothesis with a level of significance  $\alpha = 0.05$  and conclude that APDDE does obtain better optimum values than EPSDE, JADE, MDE, CODE and MDESELF. The smallest level of significance that results in the rejection of the null hypothesis,  $p$  value  $P(T_I)$  singular tail, is 0.023789017364247. So is the same with SD of the optimum values.

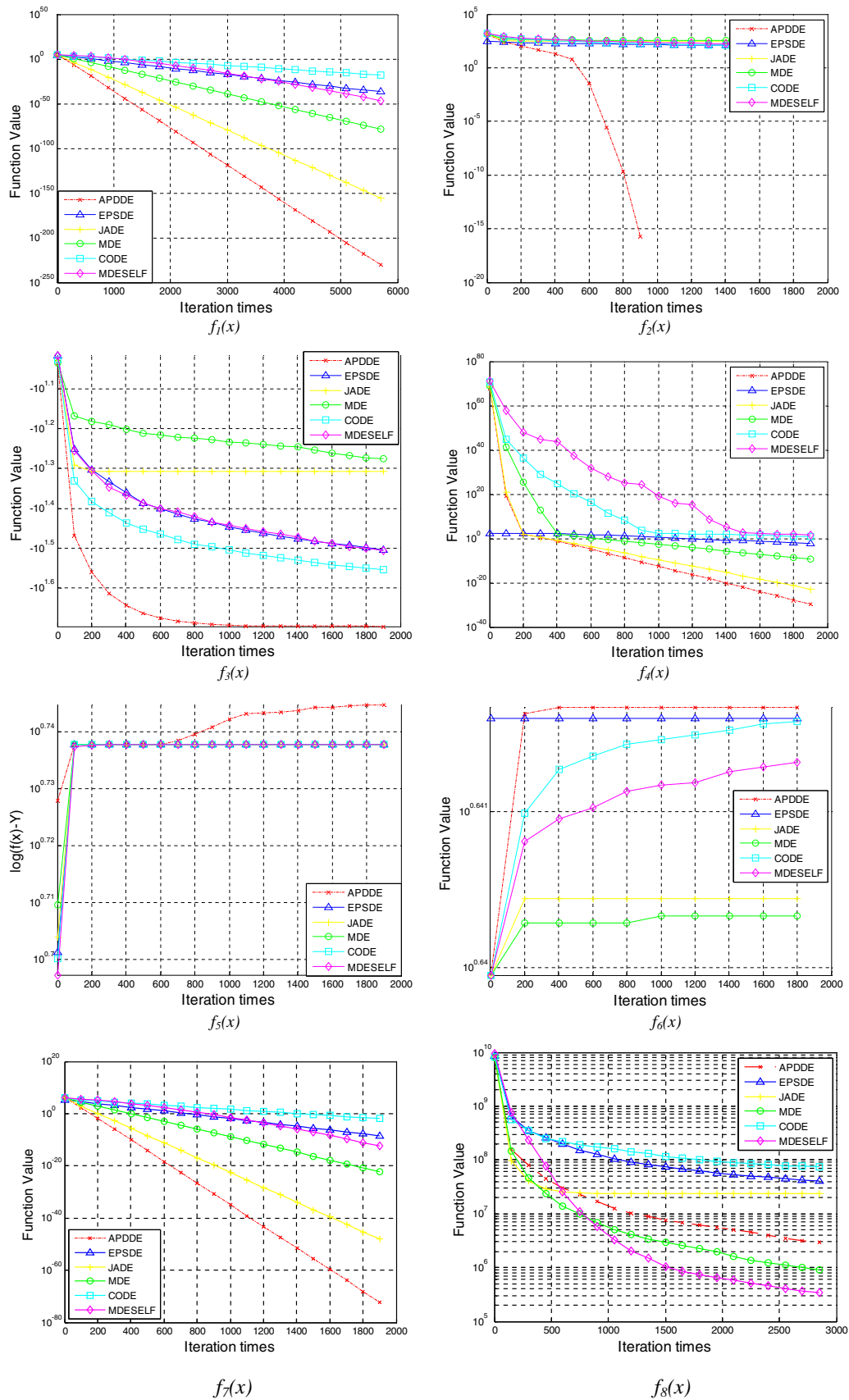


Fig. 8 Convergence curve of the 20 test functions in different algorithms

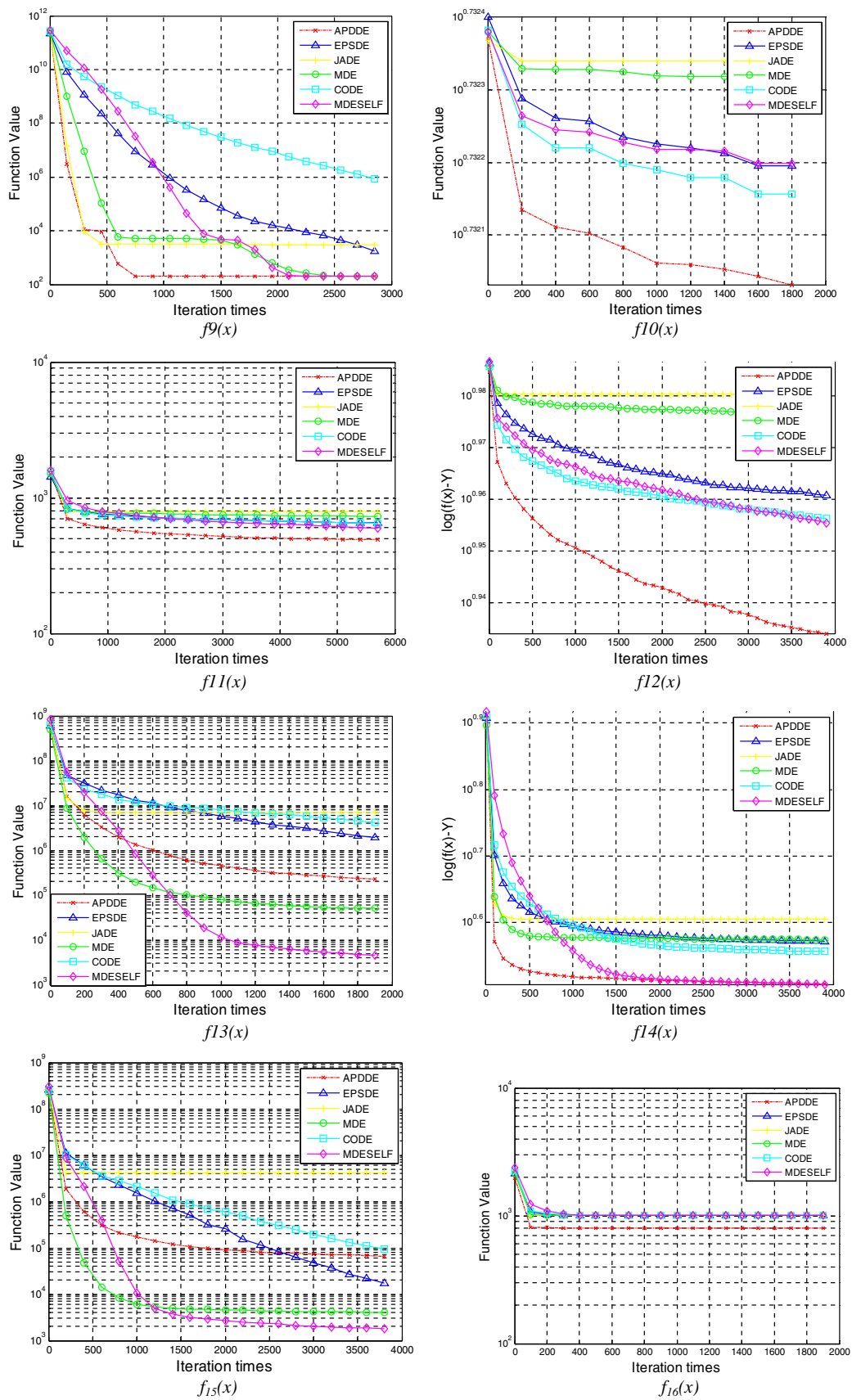


Fig. 8 continued



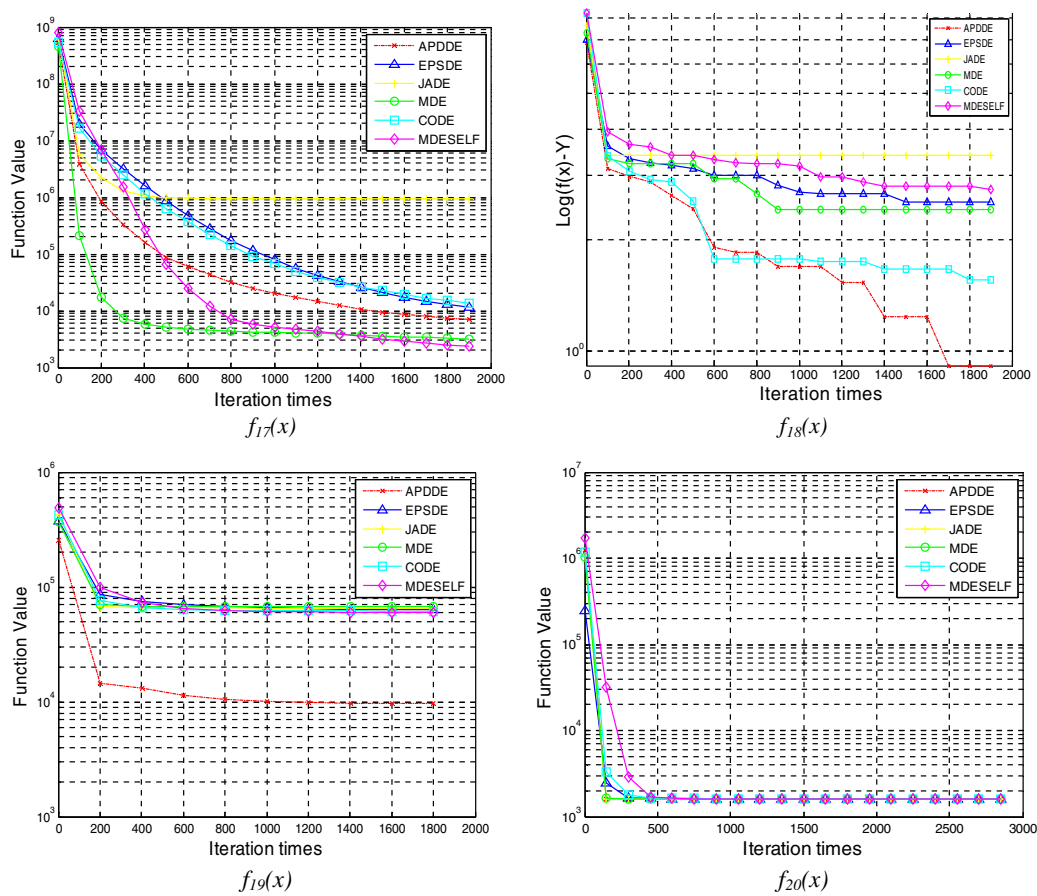


Fig. 8 continued

Table 5 T test: matched-pairs mean test of  $f_1(x)$  function for APDDE and EPSDE

	APDDE	EPSDE
Mean	9.651322E-244	1.101512E-39
Variance	8.251834E-231	3.440783E-78
Observed value	25	25
Poisson correlation coefficient	0.956892310702165	
Assume that mean difference	0	
$df$	24	
$t$ Stat	-1.09516734902491	
$P(T \leq t)$ singular tail	0.023789017364247	
$t$ (single-tailed critical)	1.6893726748929	
$P(T \leq t)$ two-tailed	0.034444862345631	
$t$ (two-tailed critical)	2.02083490575662	

Under Wilcoxon matched-pairs signed ranks test, the SD of optimum values of APDDE is significantly smaller than that of EPSDE with  $p$  value 0.023789017364247, too. If  $p$  value is less than 0.05, then the APDDE method has remarkable difference which is indicated by plus '+' in column, otherwise minus '-' is used, as shown in Table 7.

As far as others (2-20 functions),  $T$  test matched-pairs standard deviation analysis (as shown in Table 8) and

Wilcoxon matched-pairs signed ranks test (as shown in Table 9) is summarized, respectively.

In Table 9, the percentage of the '+' during all the results is the effectiveness of the APDDE in handling the test problems; for example, as to the test function nos. 2, 3, 6, 7, 8, 9, 13, 14, 17, 19, there are no values (numbers of minus '-') greater than 0.05, and ten (numbers of plus '+') are less than 0.05, which shows that the effectiveness of the APDDE

**Table 6** *F* test: matched-pairs standard variance test of  $f_1(x)$  function for APDDE and EPSDE

	APDDE	EPSDE
Mean	9.651322E-244	1.101512E-39
Std	7.569811E-243	1.854935E-39
Observed value	25	25
<i>df</i>	24	
<i>F</i>	8.414174930129825E-10	
$P(T \leq t)$ singular tail	0	
<i>F</i> (single-tailed critical)	0.927534650575722	

**Table 7** Wilcoxon matched-pairs signed ranks test of  $f_1(x)$  function for variants

Algorithms	EPSDE	JADE	MDE	CODE	MDESELF
APDDE	+/+	+/+	+/+	+/+	-/+

method in handling those relative problems is equivalent to about 100%. The result of *T* test, *F* test and Wilcoxon matched-pairs signed ranks test on APDDE is shown in Table 9. All the percentages are over (176+/190) 92.63%, which means that the experimental data are reliable and universal; the APDDE really has good performance in reducing the computational complexity and calculating cost. It is clear from Fig. 8 that, for most test problems, convergence speed of APDDE is faster among the considered algorithm.

The results of APDDE have also been compared with the results of the standard PSO, SFLA and MSFLA presented in Li et al. (2012), the basic ABC, ABCMR, dABC algorithms in Kiran and Findik (2015), ETLBO (elite teaching-learning-based optimization), CoBiDE (differential evolution based on covariance matrix learning and bimodal distribution parameter setting), GABC (Gbest-guided artificial bee colony algorithm), EDS (elite differential search algorithm) in Liu et al. (2015a).

Table 10 shows that the effect of APDDE is better than that of other algorithms; especially for *F2* function, APDDE can achieve the best value by 100% probability, and convergence of *F1* function is far higher than that of other intelligent algorithms.

### 5.4 Convergence analysis of the APDDE

APDDE contains not only inner iterative operation, but also information exchange among subpopulations; in order to analyze the convergence, we first analyze one subgroup.

**Definition 1** Assume  $X(t)$  is the  $t$ th iteration state of the population and  $F^*$  is the global optimal fitness; if the formula (14) is established, then the APDDE is global convergence.

$$\lim_{t \rightarrow \infty} P \{ f(PX(t)) = F^* \} = 1 \tag{14}$$

**Definition 2** For  $\forall x_i \in X, \forall x_j \in X$ , in APDDE’s iteration process, individual state changes from  $x_i$  to  $x_j$  by one step which is denoted as  $T_x(x_i) = x_j \circ$

**Theorem 1** For population state sequence  $\{X(t) : t \geq 0\}$ ,  $M$  is a closed set in population state space of  $X$ , which is optimal individual vector state set.

Assume  $\forall x_j \notin M, \forall x_i \in M$ , for any iteration times  $l, l \geq 1$ , according to Chapman–Kolmogorov:

$$P^l_{x_i, x_j} = \sum_{x_{r_1} \in X} \cdots \sum_{x_{r_2} \in X} P(T_x(x_i) = x_{r_1}) P(T_x(x_{r_1}) = x_{r_2}) \cdots P(T_x(x_{r_{l-1}}) = x_j) \tag{15}$$

$P^l_{x_i, x_j}$  is the probability of population state from  $x_i$  to  $x_j$  by  $l$  iteration. Since each expansion probability is 0 in formula (15),  $P^l_{x_i, x_j} = 0$ . We can conclude that  $M$  is a closed set of  $X$ .

**Theorem 2** Population state space  $X$  of the differential evolution does not have non-empty closed set  $G$ , so  $M \cap G \neq \emptyset$ .

**Theorem 3** Assume Markov chain has a non-empty closed set  $C$  and another non-empty closed set  $D$ , which makes  $C \cap D \neq \emptyset$ , does not exist. Then, when  $j \in C : \lim_{n \rightarrow \infty} P(Y_n = j) = \pi_j$ , and  $j \notin C, \lim_{n \rightarrow \infty} P(Y_n = j) = 0$ .

**Theorem 4** when the iteration in subgroup tends to infinity, the population state sequence will enter the optimal state sequence set  $M$ .

The conclusion is proved by the above Theorems 1, 2 and 3.

## 6 Conclusion

In nature, there are many things worthy of human beings to learn from the social life of ants to the foraging behavior of bees. Many natural phenomena inspire people to

**Table 8** *T* test and *F* test of Func.# 2–20 test functions

Func.#	Algorithms	Test	EPSDE	JADE	MDE	CODE	MDESELF
$f_2(x)$	APDDE	<i>T</i> test	3.74317E−05	1.32024E−09	1.25894E−06	1.37503E−12	1.16616E−13
		<i>F</i> test	0	0	0	0	0
$f_3(x)$	APDDE	<i>T</i> test	6.14242E−35	3.39677E−36	2.45557E−31	1.96756E−32	1.91993E−31
		<i>F</i> test	1.81819E−54	2.54357E−20	3.77426E−15	2.72313E−54	7.3367E−58
$f_4(x)$	APDDE	<i>T</i> test	0.059680976	0.00110574	0.004028324	2.99861E−08	0.002621721
		<i>F</i> test	1.1288E−289	4.19738E−70	4.632E−212	0	0
$f_5(x)$	APDDE	<i>T</i> test	2.47134E−10	2.47134E−10	2.47134E−10	2.47134E−10	2.47134E−10
		<i>F</i> test	0.092069658	0.001040762	2.00124E−09	2.27333E−11	7.74943E−08
$f_6(x)$	APDDE	<i>T</i> test	0.007050249	1.54859E−11	5.61742E−14	3.64418E−10	2.88344E−10
		<i>F</i> test	0	0	0	0	0
$f_7(x)$	APDDE	<i>T</i> test	5.30837E−05	9.17083E−05	0.000319104	2.99942E−14	0.008042646
		<i>F</i> test	0	0	0	0	0
$f_8(x)$	APDDE	<i>T</i> test	3.8195E−08	0.000103842	0.005663932	9.10608E−08	0.000453872
		<i>F</i> test	0.000495329	5.50769E−06	0.002935199	2.48346E−06	7.42254E−07
$f_9(x)$	APDDE	<i>T</i> test	0.008394024	0.006937064	0.031129351	3.48392E−07	0
		<i>F</i> test	5.13993E−06	1.99554E−10	0.022965978	1.96147E−30	0.85144E−22
$f_{10}(x)$	APDDE	<i>T</i> test	4.89724E−10	9.08862E−12	1.34147E−11	9.61428E−09	6.50418E−10
		<i>F</i> test	0.050972121	0.019728135	0.012162442	0.059732206	0.02476969
$f_{11}(x)$	APDDE	<i>T</i> test	4.66784E−25	9.26074E−27	2.08115E−16	5.10709E−25	5.14051E−19
		<i>F</i> test	0.875070593	0.000182211	4.70295E−12	0.486931021	0.101822692
$f_{12}(x)$	APDDE	<i>T</i> test	4.09805E−26	9.84342E−31	7.46552E−34	3.15963E−25	1.23689E−20
		<i>F</i> test	0.132235418	1.15555E−25	5.463E−33	0.624425263	0.021570809
$f_{13}(x)$	APDDE	<i>T</i> test	1.43799E−12	2.15784E−12	4.1272E−07	4.02529E−15	6.89777E−08
		<i>F</i> test	3.6844E−10	1.14439E−70	1.0078E−113	1.33073E−15	9.4001E−39
$f_{14}(x)$	APDDE	<i>T</i> test	1.3151E−11	5.39199E−14	1.82584E−19	7.62916E−06	0.03520615
		<i>F</i> test	1.60569E−12	1.40235E−24	1.73592E−37	6.43674E−19	6.81464E−20
$f_{15}(x)$	APDDE	<i>T</i> test	0.00011659	2.49543E−10	1.17937E−05	0.100891921	6.82971E−06
		<i>F</i> test	5.38708E−18	9.64984E−55	5.3529E−131	1.13586E−05	3.59258E−50
$f_{16}(x)$	APDDE	<i>T</i> test	1.75387E−59	6.92291E−58	2.86892E−59	5.43448E−59	1.04226E−58
		<i>F</i> test	0.340393904	5.88957E−61	3.00404E−65	0.020101165	0.543605787
$f_{17}(x)$	APDDE	<i>T</i> test	0.000284027	1.30735E−07	7.12828E−05	3.82864E−07	6.24221E−06
		<i>F</i> test	1.65417E−05	9.4566E−13	9.97484E−85	3.15158E−06	3.77856E−19
$f_{18}(x)$	APDDE	<i>T</i> test	0.476546547	9.87149E−10	2.91372E−06	0.000530717	0.101561488
		<i>F</i> test	0.019028261	0.171519489	0.056496135	0.255414082	0.018072766
$f_{19}(x)$	APDDE	<i>T</i> test	1.38964E−11	2.5627E−12	1.16689E−13	1.36568E−12	1.3155E−12
		<i>F</i> test	0.000372511	1.71853E−85	3.73001E−91	1.01821E−13	1.58926E−14
$f_{20}(x)$	APDDE	<i>T</i> test	2.75727E−06	8.14332E−08	2.44442E−06	2.93688E−07	6.10492E−08
		<i>F</i> test	0.761304184	0.202069719	0.00074731	2.35386E−06	0.010361514

think. This paper generates the corresponding detection values for individual per iteration; the whole APDDE algorithm optimization process can have certain direction. In addition, in order to enrich the diversity of the population, the whole population can be divided into a predefined number of groups. The individuals of each group are attracted

by the best vector of their own group. After integrating the values of sniffer into two mutation strategy to produce offspring population, the experimental studies in this paper show that the proposed APDDE algorithm improves the existing performance of other algorithms compared to some same benchmark.

**Table 9** Wilcoxon matched-pairs signed ranks test of 19 test functions

Func.#	Algorithms	EPSDE	JADE	MDE	CODE	MDESELF
$f_2(x)$	APDDE	+/+	+/+	+/+	+/+	+/+
$f_3(x)$	APDDE	+/+	+/+	+/+	+/+	+/+
$f_4(x)$	APDDE	-/+	+/+	+/+	+/+	+/+
$f_5(x)$	APDDE	+/-	+/+	+/+	+/+	+/+
$f_6(x)$	APDDE	+/+	+/+	+/+	+/+	+/+
$f_7(x)$	APDDE	+/+	+/+	+/+	+/+	+/+
$f_8(x)$	APDDE	+/+	+/+	-/+	+/+	+/+
$f_9(x)$	APDDE	+/+	+/+	+/+	+/+	+/+
$f_{10}(x)$	APDDE	+/-	+/+	+/+	+/-	+/+
$f_{11}(x)$	APDDE	+/-	+/+	+/+	+/+	+/-
$f_{12}(x)$	APDDE	+/-	+/+	+/+	+/-	+/+
$f_{13}(x)$	APDDE	+/+	+/+	+/+	+/+	+/+
$f_{14}(x)$	APDDE	+/+	+/+	+/+	+/+	+/+
$f_{15}(x)$	APDDE	+/+	+/+	+/+	-/+	+/+
$f_{16}(x)$	APDDE	+/-	+/+	+/+	+/+	+/-
$f_{17}(x)$	APDDE	+/+	+/+	+/+	+/+	+/+
$f_{18}(x)$	APDDE	-/+	+/-	+/+	+/+	-/-
$f_{19}(x)$	APDDE	+/+	+/+	+/+	+/+	+/+
$f_{20}(x)$	APDDE	+/-	+/-	+/+	+/+	+/+

**Table 10** Comparison results of function optimization

Function	Dim	Algorithm	Avg	Std
$f_6(x) = -20 \exp\left(-0.2 \sqrt{\frac{1}{N} \sum_{i=1}^N x_i^2}\right) - \exp\left(\frac{1}{N} \sum_{i=1}^N \cos(2\pi x_i)\right) + 20 + e$	50	EDS	2.1310e-03	2.0044e-03
		CoBiDE	5.2226e-07	1.1320e-06
		ETLBO	4.4409e-15	1.5979e-15
		ABC	2.24E-04	1.06E-04
		dABC	1.60E-06	7.61E-07
		GABC	1.3416e-06	1.6696e-06
		ABC (MR = 0.3)	2.46E-05	6.48E-06
		$f_2(x) = \sum_{i=1}^N (x_i^2 - 10 \cos(2\pi x_i) + 10)$	50	PSO
SFLA	1.96			- (FFEs)
MSFLA	0.0002			9.9E+06 (FFEs)
EDS	2.0988e+01			4.6707e+00
COBiDE	9.0290e+01			2.4236e+01
ETLBO	1.9280e+01			1.5957e+01
ABC	1.86E+00			1.30E+00
dABC	1.62E+00			1.36E+00
GABC	7.0737e-01			7.3535e-01
ABCMR (MR = 0.3)	2.09E+02			1.26E+01
$f_1(x) = \sum_{i=1}^N x_i^2$	50	ABC	4.75E-08	6.48E-08
		dABC	5.32E-13	6.20E-13
		ABCMR (MR = 0.3)	7.76E-09	3.67E-09
		ABCMR (MR = 0.7)	1.33E-02	4.60E-03

**Acknowledgements** Financial supports from the National Natural Science Foundation of China (No. 61572074) and the 2012 Ladder Plan Project of Beijing Key Laboratory of Knowledge Engineering for Materials Science (No. Z121101002812005) are highly appreciated.

#### Compliance with ethical standards

**Conflicts of interest** All authors of the paper declare that there is no conflict of interest each other.

**Human and animal rights** This article does not contain any studies with human participants or animals performed by any of the authors.

## References

- Ali M, Pant M, Abraham A (2013) Improving differential evolution algorithm by synergizing different improvement mechanisms. *ACM Trans Auton Adapt Syst* 7(2):20–52
- Brest J, Greiner S, Boskovic B, Mernik M, Zumer V (2009) Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. *IEEE Trans Evol Comput* 10:646–657
- Das S, Suganthan P (2011) Differential evolution: a survey of the state-of-the-art. *IEEE Trans Evol Comput* 15(1):4–31
- Fan Q, Yan X (2015) Self-adaptive differential evolution algorithm with discrete mutation control parameters. *Expert Syst Appl* 42(3):1551–1572
- Feoktistov V, Janaqi S (2004) Generalization of the strategies in differential evolution. In: *Proceedings of the 18th IPDPS*, p 165a
- Gao ZQ, Pan ZB, Gao JH (2014) A new highly efficient differential evolution scheme and its application to waveform inversion. *IEEE Geosci Remote Sens Lett* 11(10):1702–1706
- Giagkiozis I, Purshouse RC, Fleming PJ (2015) An overview of population-based algorithms for multi-objective optimisation. *Int J Syst Sci* 46:1572–1599
- Guo JGW-P, Hou F, Wang C, Cai Y-Q (2015) Adaptive differential evolution with directional strategy and cloud model. *Appl Intell* 42(2):369–388
- Hu G, Qiao P (2016) High dimensional differential evolution algorithm based on cloud cluster and its application in network security situation prediction. *J Jilin Univ Eng Technol Ed* 46(2):568–577
- Kıran MS, Findik O (2015) A directed Artificial Bee Colony algorithm. *Appl Soft Comput* 26:454–462
- Kiranyaz S, Pulkkinen J, Gabbouj M (2011) Multi-dimensional particle swarm optimization in dynamic environments. *Expert Syst Appl* 38(3):2212–2223
- Li X, Yin M (2014) Modified differential evolution with self-adaptive parameters method. *J Comb Optim* 29(111):22
- Li X, Luo J, Chen M-R, Wang N (2012) An improved shuffled frog-leaping algorithm with extremal optimisation for continuous optimization. *Inf Sci* 192:143–151
- Liang JJ, Qu BY, Suganthan PN, Chen Q (2015) Problem definitions and evaluation criteria for the CEC 2015 competition on learning-based real-parameter single objective optimization. Zhengzhou University, Zhengzhou China And Technical Report, Nanyang Technological University, Singapore
- Liu J, Zhu H, Ma Q, Zhang L, Honglei X (2015a) An Artificial Bee Colony algorithm with guide of global & local optima and asynchronous scaling factors for numerical optimization. *Appl Soft Comput* 37:608–618
- Liu R, Fan J, Jiao L (2015b) Integration of improved predictive model and adaptive differential evolution based dynamic multi-objective evolutionary optimization algorithm. *Appl Intell* 0924-669x
- Liu Z, Xu Y, Wang FM (2015c) Application of Modified differential evolution algorithm to non-linear MPC. *J Beijing Univ Technol* 41(5):680–685
- Mallipeddi R, Suganthan P, Pan Q, Tasgetiren M (2011) Differential evolution algorithm with ensemble of parameters and mutation strategies. *Appl Soft Comput* 11:1679–1696
- Poikolainen I, Neri F, Caraffini F (2015) Cluster-based population initialization for differential evolution frameworks. *Inf Sci* 297:216–235
- Price KV (1997) Differential evolution vs. the functions of the 2nd ICEO. In: *Proceedings of the IEEE International Conference on Evolutionary Computing*, pp 153–157
- Ronkkonen J, Kukkonen S, Price KV (2005) Real-parameter optimization with differential evolution. In: *IEEE congress on evolutionary computation*, pp 506–513
- Sharma H, Bansal JC, Arya KV (2015) Self balanced differential evolution. *J Comput Sci* 5(2):312–323
- Storn R, Price KV (1997) Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. *J Glob Opt* 11(4):341–359
- Wang Y, Cai ZX, Zhang QF (2011) Differential evolution with composite trail vector generation strategies and control parameters. *IEEE Trans Evol Comput* 15(1):55–66
- Yang M, Cai Z, Li C (2013) An improved adaptive differential evolution algorithm with population adaptation. In: *GECCO'13 Proceedings of the 15th annual conference on Genetic and evolutionary computation*, pp 145–152
- Yi W, Gao L, Li X, Zhou Y (2015) A new differential evolution algorithm with a hybrid mutation operator and self-adapting control parameters for global optimization problems. *Appl Intell* 42(2):642–660
- Yuan Y, Ling Z, Gao C, Cao J (2014) Formulation and application of weight-function-based physical programming. *Eng Optim* 46(12):1628–1650
- Zhang JQ, Sanderson AC (2009) JADE: adaptive differential evolution with optional external archive. *IEEE Trans Evol Comput* 13:945–958