Journal of Intelligent & Fuzzy Systems 28 (2015) 1047–1060
DOI:10.3233/IFS-141388
IOS Press

A novel mutation differential evolution for global optimization

Xiaobing Yu[a,b,c,*], Mei Cai[b,c] and Jie Cao[a]
[a]*Collaborative Innovation Center on Forecast and Evaluation of Meteorological Disasters, Nanjing University of Information Science & Technology, Nanjing, China*
[b]*China Institute of Manufacturing Development, Nanjing University of Information Science & Technology, Nanjing, China*
[c]*School of Economics and Management, Nanjing University of Information Science & Technology, Nanjing, China*

Abstract. Differential evolution (DE) is a simple and powerful population-based evolutionary algorithm. The success of DE in solving a specific problem crucially depends on appropriately choosing generation strategies and control parameter values. A novel mutation DE (MDE) is proposed to improve generation strategy DE/best/2. Adaptive mutation is conducted to current population when the population clusters around local optima. Control parameters are adapted based on constants. The performance of MDE is extensively evaluated on eighteen benchmark functions and compares favorably with the several DE variants.

Keywords: Evolutionary algorithm, global optimization, differential evolution, DE/best/2, particle swarm optimization

1. Introduction

Evolutionary algorithms (EAs), inspired by the natural evolution of species, have been successfully applied to solve numerous optimization problems in diverse fields [1]. Differential evolution (DE) is the stochastic, population-based optimization EAs [2]. It is a simple yet powerful EA for global optimization introduced by Price and Storn [3]. The DE algorithm has gradually become more popular and has been used in many practical cases, mainly because it has demonstrated good convergence properties and is principally easy to understand. DE has been successfully applied in diverse fields of engineering [4–7].

The performance of the conventional DE algorithm highly depends on the chosen trial vector generation strategy and associated parameter values used. Inappropriate choice of strategies and parameters may lead to premature convergence, which has been extensively demonstrated in [8].

In the past decade, DE researchers have suggested many empirical guidelines for choosing trial vector generation strategies and their associated control parameter settings.

(1) Generation strategies

Some new generation strategies are introduced in DE, such as DE/current-to-pbest [9], DE/target-to-best/1 [10], DE/current-to-rand/1 [11]. Besides, the ensemble concept is generally applied with DE algorithm framework and used to generate better offspring. A novel method, called composite DE (CoDE) uses three trial vector generation strategies and three control parameter settings [12]. An ensemble of mutation strategies and control parameters with the DE (EPSDE) was proposed. In EPSDE, a pool of distinct mutation strategies along with a pool of values for each control parameter coexists throughout the evolution process and competes to produce offspring [13]. A self-adaptive DE (SaDE) algorithm, in which both trial vector generation strategies and their associated control parameter values are

*Corresponding author. Xiaobing Yu, School of Economics and Management, Nanjing University of Information Science & Technology, Nanjing 210044, China. E-mail: yuxiaobing@nuist.edu.cn.

gradually self-adapted by learning from their previous experiences in generating promising solutions [1].

(2) Control parameter settings

Control parameter settings can be defined as how the control parameters are changed. Different control parameter settings have been presented, mainly including adaptive parameter control and self-adaptive parameter control. Adaptive parameter control is to feedback from the evolutionary search. The control parameters are changed dynamically. Examples of this class are JADE [9] fuzzy adaptive differential evolution (FADE) [14], jDE [15], and SaNSDE [16]. Self-adaptive parameter control is used to conduct the self adaptation of control parameters. The control parameters are directly associated with individuals and undergo mutation and recombination/crossover. Since better parameter values tend to generate individuals that are more likely to survive, these values can be propagated to more offspring. The DESAP in [17] belong to this category. Adaptive or self-adaptive parameter control, if well designed, can enhance the robustness of an algorithm by dynamically adapting the parameters to the characteristic of different fitness landscapes.

In a word, DE researchers have suggested many empirical guidelines for choosing trial vector generation strategies and control parameter settings during the last decade. It has been clear that these efforts are useful for optimization. Undoubtedly, these experiences are very useful for improving the performance of DE. We have observed, above algorithms do not judge whether population is trapped in the local optima. This motivates us to study whether the DE performance can be improved by evaluating current individual. In this paper, an adaptive mutation is conducted to current individual when the individual clusters around local optima. Control parameters are adapted based on constants. With the efforts, the proposed algorithm has the ability to jump out of the local optima.

2. Conventional DE

DE is proposed by Storn and Price [3, 7]. It is an effective, robust and simple global optimization algorithm. According to frequently reported experimental studies, DE has shown better performance than many other EAs in terms of convergence speed and robustness over several benchmark functions and real-world problems [18]. The Initialization and DE operators are explained briefly as following [1, 19].

DE starts with a population of *NP* *D*-dimensional candidate solutions which may be represented as $X_{i,G}(i=1,2,\ldots,NP)=\{x_{i,G}^1,\ x_{i,G}^2,\ldots,x_{i,G}^D\}$, where index i denotes the *ith* individual of the population, G denotes the generation to which the population belongs and D is the dimension of the population.

The initial population should try to cover the entire search space as much as possible by uniformly randomizing individuals within the search space constrained by the minimum $X_{\min}=\{x_{\min}^1,x_{\min}^2,\ldots,x_{\min}^D\}$ and maximum $X_{\max}=\{x_{\max}^1,x_{\max}^2,\ldots,x_{\max}^D\}$ bounds. Thus, the initial population can be described as the following: $x_{i,0}=x_{\min}+rand(0,1)\times(x_{\max}-x_{\min})$ where $rand(0,1)\in[0,1]$ is a uniformly distributed random variable [19, 20]. The main procedure of DE is as following [1].

(1) Mutation

After initialization, DE utilizes the mutation operation to generate a trial vector $V_{i,G}=\{v_{i,G}^1, v_{i,G}^2,\ldots,v_{i,G}^D\}$ with respect to each individual in the current population. $V_{i,G}$ can be produced by certain mutation strategy. For example, the five most frequently mutation strategies implemented in the DE are listed as following:

DE/rand/1:

$$V_{i,G}=X_{r_1^i,G}+F.(X_{r_2^i,G}-X_{r_3^i,G}) \tag{1}$$

DE/best/1:

$$V_{i,G}=X_{best,G}+F.(X_{r_1^i,G}-X_{r_2^i,G}) \tag{2}$$

DE/rand-to-best/1:

$$V_{i,G}=X_{i,G}+F.(X_{best,G}-X_{i,G})+F.(X_{r_1^i,G}-X_{r_2^i,G}) \tag{3}$$

DE/best/2:

$$V_{i,G}=X_{best,G}+F.(X_{r_1^i,G}-X_{r_2^i,G})+F.(X_{r_3^i,G}-X_{r_4^i,G}) \tag{4}$$

DE/rand/2:

$$V_{i,G}=X_{r_1^i,G}+F.(X_{r_2^i,G}-X_{r_3^i,G})+F.(X_{r_4^i,G}-X_{r_5^i,G}) \tag{5}$$

The indices r_1^i, r_2^i, r_3^i, r_4^i, r_5^i are mutually exclusive integers randomly generated within the range [0, 1], which are also different from the index i [1]. F is the mutation scale factor which is used in controlling the amplification of the differential variation [19].

(2) Crossover

The crossover operation is introduced to increase the diversity of the target vectors. After the mutation phase, the crossover operation is applied to $V_{i,G}=\{v_{i,G}^1, v_{i,G}^2,\ldots,v_{i,G}^D\}$ and $X_{i,G}=\{X_{i,G}^1,\ X_{i,G}^2,\ldots,\ X_{i,G}^D\}$ to generate a trial vector $U_{i,G} = \{u^1_{i,G},\ u^2_{i,G}, \ldots,\ u^D_{i,G}\}$ as follows:

$$u^j_{i,G} = \begin{cases} v^j_{i,G}, ifrand_j[0,1) \le CRor(j = j_{rand}) \\ x^j_{i,G}, others \end{cases} \tag{6}$$

$CR \in [0, 1]$ is the crossover constant, which has to be determined by the user. $j_{rand} \in [1, D]$ is a randomly chosen index which ensures that the trial vector $U_{i,G}$ will differ from $X_{i,G}$ by at least one parameter.

The performance of the DE algorithm is very sensitive to F and CR. Adaptive approach is usually used to generate parameter F and CR. Parameters $F_{i,G+1}$ and $CR_{i,G+1}$ of jDE [15] are calculated as

$$F_{i,G+1} = \begin{cases} F_l + rand_1 \times F_u & if\ rand_2 < \tau_1 \\ F_{i,G} & otherwise \end{cases} \tag{7}$$

$$CR_{i,G+1} = \begin{cases} rand_3 & if\ rand_4 < \tau_2 \\ CR_{i,G} & otherwise \end{cases} \tag{8}$$

and they produce factors F and CR in a new parent vector. $rand_j(j \in \{1,\ 2,\ 3,\ 4\})$ are uniform random values within the range [0,1]. τ_1 and τ_2 represent probabilities to adjust factors F and CR respectively. Experimental results suggest that jDE performs remarkably better than the classic DE/rand/1/bin, FADE algorithm [15].

(3)Selection

If the generated trial vector $u^j_{i,\ G}$ exceeds the corresponding upper and lower bounds, we randomly and uniformly reinitialize it within the search range. Then the fitness values of all trial vectors are evaluated [20, 21].

After that, a greedy selection scheme is employed

$$x_{i,G+1} = \begin{cases} u_{i,G} & if\ f(u_{i,G}) \le f(x_{i,G}) \\ x_{i,G} & otherwise \end{cases} \tag{9}$$

If the trial vector $u_{i,G}$ yields a better cost function value than $x_{i,G}$, the $u_{i,G}$ will replace $x_{i,G}$ and enter the population of the next generation; otherwise, the old value $x_{i,G}$ is retained.

3. Mutation DE

3.1. Mutation DE(MDE)

DE/rand/1 is the first mutation strategy developed for DE [22, 23], and is said to be the most successful and widely used scheme in the literature [24]. However, [25] indicates that DE/best/2 may have some advantages over DE/rand/1, and [26] favors DE/best/k for most technical problems investigated. Also, the authors of [9, 27] argue that the incorporation of best solution information is beneficial. Compared to DE/rand/1 and DE/rand/2, DE/best/2 benefits from their fast convergence by incorporating best solution information in the evolutionary search. However, the best solution information may also cause problems such as premature convergence due to the resultant reduced population diversity [9]. In order to enhance the performance of DE, the proposed algorithm adds mutation mechanism on DE/best/2 to improve generation strategy.

When an individual discovers a current optima position, others will draw together to the individual. If the position is the local optima, the algorithm will be convergence and clustered in local optima. The premature may appear. Suppose that the population size of MDE is NP, the fitness value of ith individual is f_i and the average fitness value is f_{avg}. The convergence degree is defined as following:

$$d = \sqrt{\sum_{i=1}^{NP} (\frac{f_i - f_{avg}}{dev})^2} \tag{10}$$

Where

$$dev = \begin{cases} \max(f_i - f_{avg}) & \max(f_i - f_{avg}) \ne 0 \\ 1 & \max(f_i - f_{avg}) = 0 \end{cases} \tag{11}$$

The parameter d reflects the convergence degree. When the parameter d is large, the algorithm is in random search. On the other hand, the algorithm will be convergence and premature maybe occur. In order to evaluate the parameter d, d_c is given as following, where p is the mutation probability.

$$p = \begin{cases} k & d < d_c \\ 0 & others \end{cases} \tag{12}$$

Generally, $d_c \in [1.0, 2.5]$. If the parameter d is less than d_c, the mutation probability p is equal to $k \in [0.2, 0.5]$. The mutation is as follows:

$$X_{best,G} = (1 + 0.5\eta)X_{best,G} \tag{13}$$

where the parameter η obeys Gauss(0,1) distribution. The Gaussian random number parameter η is unique at each dimension of $X_{best,G}$.

The above discussion is generation strategy. Control parameter settings follow the research [15], which has been approved. The efforts improve the convergence rate of DE and maintain diversity of the population.

The main procedure of MDE is presented as following. The comparison of MDE with DE variants are presented in Table 1.

Step 1 Initialize parameters: Convergence evaluation parameter d_c; the mutation probability k.

Step 2 generation counter $G=0$ and randomly initialize a population of NP individuals $P_G = \{X_{1,G}, \ldots, X_{NP,G}\}$ with $X_{i,G} = \{x_{i,G}^1, \ldots, x_{i,G}^D\}$, $i = 1, \ldots, NP$ uniformly distributed in the range $[X_{\min}, X_{\max}]$, where $X_{\min} = \{x_{\min}^1,\ x_{\min}^2, \ldots,\ x_{\min}^D\}$ and $X_{\max} = \{x_{\max}^1,\ x_{\max}^2, \ldots,\ x_{\max}^D\}$.

Step 3 Evaluation the population and Identify the best position.

```
Step 4 While stopping criterion is not satisfied for $i = 1$ to the $NP$ do
      Update $F$ and $R$ according to Equations (7, 8);
      Generate $V_{i,G}$ using the Equation (4)
      Generate the trial vector $U_{i,G}$ by the Equation (6)
  End for
```

Step 4.1 Randomly reinitialize the trial vector $U_{i,G}$ within the search space if any variable is outside its boundaries.

```
Step 4.2 Selection
  for $i = 1$ to $NP$
      Evaluate the trial vector $U_{i,G}$
      If $f(U_{i,G}) \leq f(X_{i,G})$
          $X_{i,G+1} = U_{i,G},\ f(X_{i,G+1}) = f(U_{i,G})$
          $P_{i,pbest} = U_{i,G},\ f(P_{i,pbest}) = f(U_{i,G})$
    If $f(U_{i,G}) < f(X_{best,G})$
          $X_{best,G} = U_{i,G},\ f(X_{best,G}) = f(U_{i,G})$
              $p_g = U_{i,G},\ f(p_g) = f(U_{i,G})$
              End if
      End if
  End for
```

```
Step 4.3 Calculate parameter $d$ using Equation (10)
  If parameter meets the requirement of Equation (12)
      Generate a random number $rand$ in (0,1).
          If $rand$ is less than $k$
              Update $X_{best,G}$ using Equation (13)
          End if
      End if
```

Step 4.4 Increment the generation count $G = G+1$;

Step 5 End while

Table 1
Comparison of MDE with jDE, SaDE, JADE, CoDE and EPSDE

Algorithm	Comparison
jDE	The ranges of *F* and *CR* values are [0.1, 1.0] and [0, 1], respectively. The *F* and *CR* values are adapted based on two constants, which are set to 0.1.
SaDE	The *F* values are randomly generated with a mean and standard deviation of 0.5 and 0.3, respectively. The mutation strategy and the parameter *CR* are self-adapted based on their previous performance.
JADE	JADE implements a new mutation strategy "DE/current-to-pbest" with optional external archive. In every generation, the *CR* value corresponding to each individual is randomly initialized with a normal distribution of mean u_{CR} and standard deviation 0.1, while the *F* value is randomly initialized with a Cauchy distribution with a location parameter u_F and standard deviation 0.1. After every generation, the u_{CR} and u_F are updated respectively.
CoDE	CoDE uses three trial vector generation strategies and three control parameter settings: rand/1/bin, rand/2/bin, current-to-rand/1.
EPSDE	Each member in the initial population is assigned a mutation strategy and parameter values randomly selected from the respective pools. The mutation strategy and parameter values producing better offspring survive while those fail to produce better offspring are reinitialized.
MDE	MDE uses mutation strategy DE/best/2 and the control parameter settings are as same as jDE. The best solution information may also cause premature convergence due to the resultant reduced population diversity. Adaptive mutation is conducted to improve population diversity when the population clusters around local optima.

The most important parameters for the proposed algorithm are convergence evaluation parameter d_c and mutation probability k. In the following, we will make an integrated analysis of the key parameters by comparing the performance of the MDE in the optimization of several representative functions.

4. Numerical experiments and results

4.1. Benchmark functions and algorithm configuration

As we wish to test the MDE on diverse testing functions and our main objective is to improve DE's performance on multimodal functions, we choose eigh-

Table 2
Benchmark configurations

Function	Name	Search space	Global optimal $f(x^*)$	x^*
f_1	Sphere	[−100, 100]	0	0
f_2	*Elliptic*	[−100, 100]	0	0
f_3	Schwefel's Problem	[−10, 10]	0	0
f_4	Rosenbrock	[−30, 30]	0	1,1,... ,1
f_5	Ackely	[−32, 32]	0	0
f_6	Griewank	[−600, 600]	0	0
f_7	Weierstrass	[−0.5, 0.5]	0	0
f_8	Rastrigrin	[−5, 5]	0	0
f_9	Noncontinuous Rastrigin	[−5, 5]	0	0
f_{10}	Rotated *Elliptic*	[−100, 100]	0	0
f_{11}	Rotated Ackely	[−32, 32]	0	0
f_{12}	Rotated Griewank	[−600, 600]	0	0
f_{13}	Rotated Weierstrass	[−0.5, 0.5]	0	0
f_{14}	Rotated Rastrigrin	[−5, 5]	0	0
f_{15}	Rotated Noncontinuous Rastrigin	[−5, 5]	0	0
f_{16}	Shifted Ackely	[−32, 32]	O*	0
f_{17}	Shifted Griewank	[−600, 600]	O*	0
f_{18}	Shifted Weierstrass	[−0.5, 0.5]	O*	0

teen benchmark functions [28–33]. All functions are tested on 30 dimensions. The benchmark functions are given in Table 2. It denotes the ranges of the variables and the value of the global optimum. According to their properties, these functions are divided into four groups: un-modal functions, basic multimodal functions, rotated multimodal functions and shifted multimodal functions. The properties and the formulas of these functions are presented below.

Group A: Non-modal and simple multimodal futions:

$$f_1 = \sum_{i=1}^{D} x_i^2 \tag{14}$$

$$f_2 = \sum_{i=1}^{D} (10^6)^{\frac{i-1}{D-1}} x_i^2 \tag{15}$$

$$f_3 = \sum_{i=1}^{D} (\sum_{j=1}^{i} x_i)^2 \tag{16}$$

$$f_4 = \sum_{i=1}^{D-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2] \tag{17}$$

Group B: Basic multimodal functions:

$$f_5 = -20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}) - \exp(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)) + 20 + e \tag{18}$$

$$f_6 = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D}\cos(\frac{x_i}{i^{1/2}}) + 1 \tag{19}$$

$$f_7 = \sum_{i=1}^{D}(\sum_{k=0}^{k_{\max}}[a^k\cos(2\pi b^k(x_i + 0.5))] - D\sum_{k=0}^{k_{\max}}[a^k\cos(2\pi b^k * 0.5)])$$
$$a = 0.5; b = 3; k_{\max} = 20 \tag{20}$$

$$f_8 = \sum_{i=1}^{D} [x_i^2 - 10\cos(2\pi x_i) + 10] \tag{21}$$

$$f_9 = \sum_{i=1}^{D}(y_i^2 - 10\cos(2\pi y_i) + 10),$$
$$y_i = \begin{cases} x_i & |x_i| < 0.5 \\ \frac{round(2x_i)}{2} & |x_i| \geq 0.5 \end{cases} \tag{22}$$

Group C: Rotated multimodal functions:

$$f_{10} = \sum_{i=1}^{D} (10^6)^{\frac{i-1}{D-1}} y_i^2,\ y = M \times x \tag{23}$$

$$f_{11} = -20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}) - \exp(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)) + 20 + e,\ y = M \times x \tag{24}$$

$$f_{12} = \frac{1}{4000}\sum_{i=1}^{D} y_i^2 - \prod_{i=1}^{D}\cos(\frac{y_i}{i^{1/2}}) + 1,\ y = M \times x \tag{25}$$

$$f_{13} = \sum(\sum[a^k \cos(2\pi b^k(y_i + 0.5))] - D$$

$$\sum_{k=0}^{k_{\max}} [a^k \cos(2\pi b^k * 0.5)]); a = 0.5; b = 3;$$

$$k_{\max} = 20, y = M \times x \quad (26)$$

$$f_{14} = \sum_{i=1}^{D} [y_i^2 - 10\cos(2\pi y_i) + 10], y = M \times x \quad (27)$$

$$f_{15}(x) = \sum_{i=1}^{D} (z_i^2 - 10\cos(2\pi z_i) + 10),$$

$$z_i = \begin{cases} y_i & |y_i| < 0.5 \\ \frac{round(2y_i)}{2} & |y_i| \geq 0.5 \end{cases}, y = M \times x \quad (28)$$

Group D: Shifted multimodal functions:

$$f_{16} = -20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} z_i^2})$$

$$-\exp(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi z_i)) + 20 + e, z = x - o \quad (29)$$

$$f_{17} = \frac{1}{4000}\sum_{i=1}^{D} z_i^2 - \prod_{i=1}^{D}\cos(\frac{z_i}{i^{1/2}}) + 1, z = x - o \quad (30)$$

$$f_{18} = \sum_{i=1}^{D}(\sum_{k=0}^{k_{\max}}[a^k \cos(2\pi b^k(z_i + 0.5))] - D$$

$$\sum_{k=0}^{k_{\max}} [a^k \cos(2\pi b^k \times 0.5)]), z = x - o; \quad (31)$$

$$a = 0.5; b = 3; k_{\max} = 20$$

Five existing DE algorithms are shown in Table 3 in detail. In our experiments, we use the same parameter settings for these five methods as in their original papers. The number of FES in all these methods is set to 300,000, as the same as in MDE. All the results are obtained from 25 independent runs.

Table 3
The reference of five DE algorithms

Algorithm	Reference
SaDE	[1]
JADE	[9]
EPSDE	[13]
CoDE	[12]
jDE	[15]

4.2. Comparisons on the solution accuracy

The performance on the solution accuracy of every DE algorithms listed in Table 4 and is compared with the each other. The results are shown in Table 4 in terms of the mean and standard deviation of the solutions obtained in the 25 independent runs by each algorithm. Boldface in the table indicates the best result among those obtained by all six contenders. Figure 1 graphically presents the comparison in terms of convergence characteristics of the evolutionary processes in solving the eighteen different functions.

1) *Un-modal functions f_1–f_4.* Clearly, EPSDE is the best among the six algorithms on these four un-modal functions. The outstanding performance of EPSDE should be due to its distinct mutation strategies along with a pool of values for each control parameter, which leads to very fast convergence. CoDE and SaDE is the second best.
2) *Basic multimodal functions f_5–f_9.* On these five testing functions, MDE is significantly better than SaDE, and EPSDE. jDE, JADE and CoDE outperforms second. Thus, MDE is the winner on these five testing functions. This can be because MDE could balance exploration and exploitation on these testing functions.
3) *Rotated multimodal functions f_{10}–f_{15}.* These testing functions are much harder than others. Overall, the performance of MDE is better than that of the five competitors. It outperforms jDE, JADE, SaDE, and EPSDE except f_{11}. It is interesting to note that for testing functions f_{11}, the performance of six algorithms is similar.
4) *Shifted multimodal functions f_{16}–f_{18}.* On these three testing functions, jDE, CoDE and JADE exhibit similar performance and outperform three other methods.

In summary, MDE is the best among the five methods in comparison on basic multimodal functions, rotated multimodal functions and shifted multimodal functions.

Table 4
Result comparisons among six algorithms on testing functions

Function		MDE	jDE	SaDE	JADE	CoDE	EPSDE
f_1	Mean	4.12E-77	1.28E-35	**0**	3.71E-122	3.59E-67	1.29E-172
	Std	4.23E-152	1.10E-70	**0**	3.40E-242	3.27E-133	0
	t-test		+	=	=	=	=
f_2	Mean	8.74E-80	1.60E-58	1.44E-128	1.52E-109	1.23E-63	**1.05E-166**
	Std	1.87E-157	5.17E-116	2.55E-255	5.73E-217	1.65E-125	**0**
	t-test		+	=	=	=	=
f_3	Mean	3.39E-17	1.42E-7	3.10E-7	3.59E-48	1.72E-16	**5.75E-42**
	Std	2.87E-32	2.87E-14	2.22E-13	3.18E-94	6.00E-32	**2.82E-82**
	t-test		+	+	=	+	=
f_4	Mean	2.46E+1	2.35E+1	1.90E+1	4.78E-1	**3.38E-11**	3.19E-1
	Std	3.49E-1	6.80E+2	3.09E+2	1.75E+0	**8.89E-21**	1.22E+0
	t-test		=	=	-	-	-
f_5	Mean	**8.88E-16**	3.84E-15	1.65E-1	3.55E-15	3.55E-15	4.69E-15
	Std	**0**	9.68E-31	1.55E-1	**0**	**0**	2.86E-30
	t-test		+	+	+	+	+
f_6	Mean	**0**	**0**	6.7E-3	**0**	**0**	3.94E-04
	Std	**0**	**0**	9.43E-5	**0**	**0**	3.89E-06
	t-test		=	+	=	=	=
f_7	Mean	**0**	**0**	5.88E-4	**0**	**0**	2.61E-1
	Std	**0**	**0**	8.37E-6	**0**	**0**	1.12E-1
	t-test		=	=	=	=	+
f_8	Mean	**0**	**0**	**0**	**0**	**0**	**0**
	Std	**0**	**0**	**0**	**0**	**0**	**0**
	t-test		=	=	=	=	=
f_9	Mean	**0**	**0**	**0**	**0**	4E-2	**0**
	Std	**0**	**0**	**0**	**0**	4E-2	**0**
	t-test		=	=	=	=	=
f_{10}	Mean	**2.59E-4**	6.35E+4	5.83E+4	4.35E+3	2.16E+4	3.85E+5
	Std	**7.95E-7**	1.46E+9	1.02E+9	3.24E+7	2.41E+8	1.82E+12
	t-test		+	+	+	+	=
f_{11}	Mean	2.09E+1	2.10E+1	2.09E+1	2.09E+1	**2.01E+1**	2.09E+1
	Std	4.3E-3	**2.2E-3**	2.4E-3	2.60E-2	1.45E-2	2.3E-3
	t-test		=	=	=	-	=
f_{12}	Mean	**0**	9.2E-3	2.23E-2	7.4E-3	6.3E-3	1.83E-2
	Std	**0**	9.77E-5	3.11E-4	6.41E-5	7.58E-5	1.86E-4
	t-test		+	+	+	+	+
f_{13}	Mean	**0**	2.75E+1	1.59E+1	2.55E+1	1.32E+1	3.38E+1
	Std	**0**	5.11E+0	3.90E+0	1.52E+0	1.15E+1	1.30E+1
	t-test		+	+	+	+	+
f_{14}	Mean	**0**	5.41E+1	2.52E+1	2.11E+1	3.37E+1	3.91E+1
	Std	**0**	7.38E+1	2.74E+1	1.75E+1	1.18E+2	1.62E+2
	t-test		+	+	+	+	+
f_{15}	Mean	**0**	7.29E+1	3.26E+1	2.71E+1	4.45E+1	4.80E+1
	Std	**0**	1.28E+2	1.11E+2	3.91E+1	1.28E+2	6.14E+2
	t-test		+	+	+	+	+
f_{16}	Mean	7.9E-1	3.55E-15	1.24E+0	4.41E-15	**3.26E-15**	1.42E+1
	Std	1.09E+1	1.05E-30	3.11E-1	2.4E-30	**9.68E-31**	5.09E+1
	t-test		=	=	=	=	+
f_{17}	Mean	1.81E-9	**0**	6.9E-3	**0**	**0**	2.96E-4
	Std	2.48E-18	**0**	1.77E-4	**0**	**0**	2.19E-6
	t-test		–	–	–	–	=
f_{18}	Mean	3.83E-1	**0**	2.89E-4	**0**	**0**	2.8E-1
	Std	1.84E-1	**0**	2.09E-6	**0**	**0**	1.31E-1
	t-test		–	–	–	–	=
	Better		9	8	6	7	7
	Same		7	8	9	7	10
	Worse		2	2	3	4	1
	Score		7	6	3	3	6

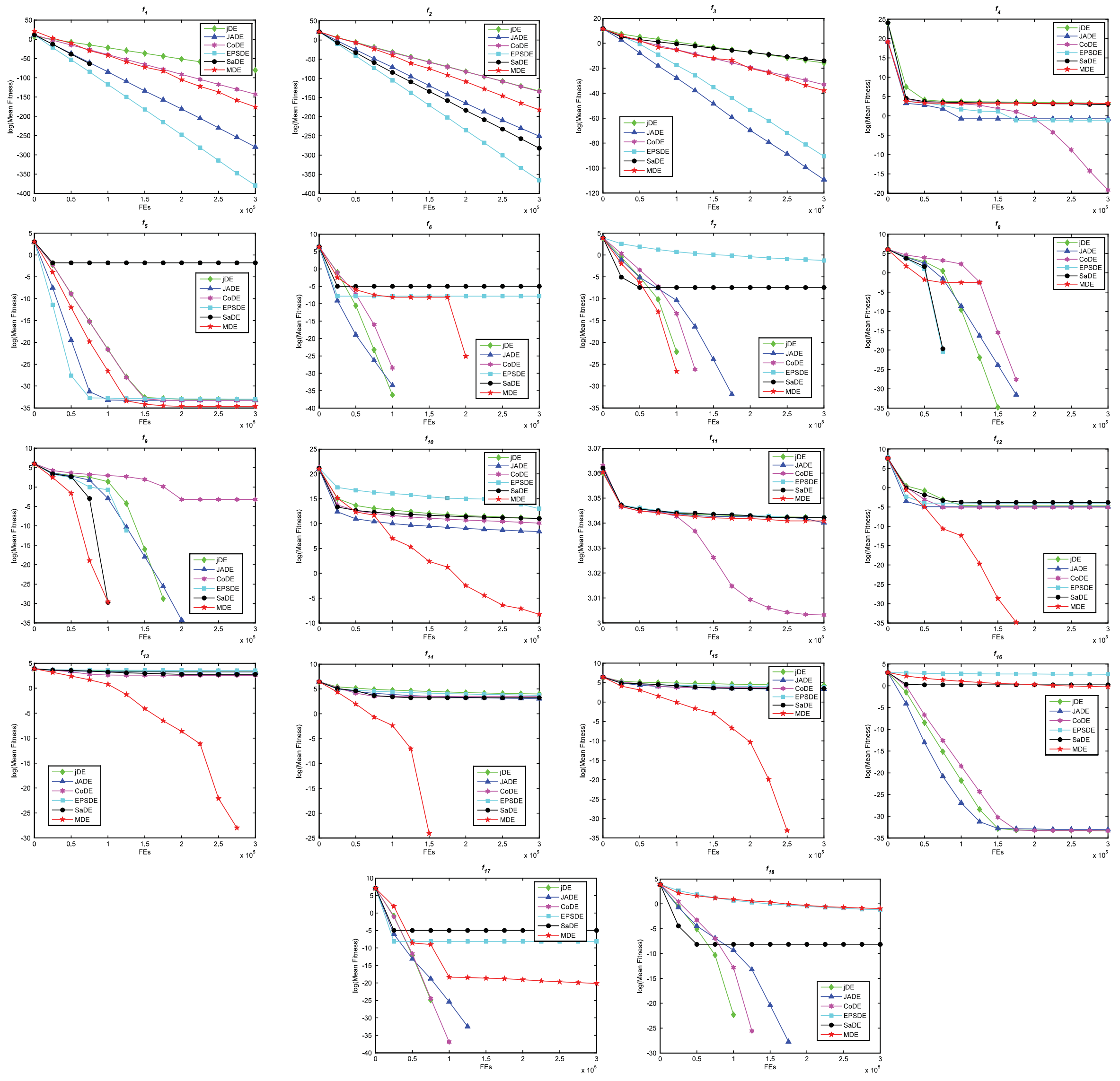

Fig. 1. Performance of the algorithms for f_1 to f_{18}.

4.3. *Comparisons on the convergence speed*

The convergent rate for achieving an acceptable solution is another key point for testing the algorithm performance. The success of an algorithm means that this algorithm can result in a function value no worse than an acceptable solution, i.e., for all problems with the number of function evaluations less than the pre specified maximum number. The success rate (SR) is calculated as the number of successful runs divided by the total number of runs. In Table 5, we summarize the SR of each algorithm and the average number of function evaluations over successful runs (FESS). An experiment is considered as successful if the best solution is found with the acceptable solution.

Table 5 shows that EPSDE needs least FESS to achieve the acceptable solution on un-modal all functions; MDE requires least FESS to obtain the acceptable solution on basic multimodal functions and rotated multimodal functions, which reveals that proposed

Table 5
Result comparisons among six algorithms on testing functions

Function		MDE	jDE	SaDE	JADE	CoDE	EPSDE
f_1	Mean FESS	4.51E+4	8.74E+4	2.8779E+4	3.0571E+4	5.27E+4	**2.12E+4**
	SR	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**
f_2	Mean FESS	5.47E+4	7.36E+4	3.37E+4	4.20E+4	6.69E+4	**2.89E+4**
	SR	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**
f_3	Mean FESS	1.39E+5	2.97E+5	2.83E+5	7.27E+4	1.79E+5	**8.74E+4**
	SR	**100%**	8%	12%	**100%**	**100%**	**100%**
f_4	Mean FESS				**1.14E+5**	2.59E+5	1.41E+5
	SR	0%	0%	0%	88%	**100%**	92%
f_5	Mean FESS	6.67E+4	8.74E+4	4.13E+4	4.73E+4	8.18E+4	**3.36E+4**
	SR	**100%**	**100%**	84%	**100%**	**100%**	**100%**
f_6	Mean FESS	6.01E+4	6.14E+4	2.85E+4	3.42E+4	5.75E+4	**2.26E+4**
	SR	**100%**	**100%**	56%	**100%**	**100%**	96%
f_7	Mean FESS	**6.47E+4**	9.20E+4	4.35E+4	1.23E+5	1.02E+5	2.02E+5
	SR	**100%**	**100%**	92%	**100%**	**100%**	28%
f_8	Mean FESS	**6.21E+4**	1.15E+5	6.79E+4	1.32E+5	1.42E+5	6.71E+4
	SR	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**
f_9	Mean FESS	**6.39E+4**	1.49E+005	8.28E+004	1.51E+005	1.90E+5	8.16E+4
	SR	**100%**	**100%**	**100%**	**100%**	96%	**100%**
f_{10}	Mean FESS	**2.40E+5**					
	SR	**36%**	0%	0%	0%	0%	0%
f_{11}	Mean FESS	1.41E+5	1.03E+5	1.46E+5	1.32E+5	**1.21E+5**	1.27E+5
	SR	64%	48%	48%	48%	**100%**	48%
f_{12}	Mean FESS	8.88E+4	1.67E+5	3.20E+5	**4.54E+4**	1.03E+5	5.95E+4
	SR	**100%**	36%	8%	44%	56%	12%
f_{13}	Mean FESS	**1.96E+5**					
	SR	**96%**	0%	0%	0%	0%	0%
f_{14}	Mean FESS	**9.72E+4**					
	SR	**100%**	0%	0%	0%	0%	0%
f_{15}	Mean FESS	**1.09E+5**					
	SR	**100%**	0%	0%	0%	0%	0%
f_{16}	Mean FESS	1.08E+5	8.74E+4	8.30E+4	**3.18E+4**	1.03E+5	1.13E+5
	SR	24%	**100%**	12%	**100%**	**100%**	16%
f_{17}	Mean FESS	1.38E+5	6.17E+4	3.07E+4	3.81E+4	6.30E+4	**2.48E+4**
	SR	**100%**	**100%**	68%	**100%**	**100%**	96%
f_{18}	Mean FESS	2.59E+5	9.18E+4	**4.54E+4**	1.29E+5	1.14E+5	2.41E+5
	SR	12%	**100%**	96%	**100%**	**100%**	8%
	Mean Realiablity	**80.63%**	62.74%	51.58%	72.63%	76.42%	57.68%

algorithm has a higher convergent rate than other algorithms. The five DE algorithms outperform MDE on shifted multimodal functions.

In summary, the MDE performs best on testing functions and has good search ability. Owing to the proposed techniques, the MDE processes capabilities of fast convergence speed, the highest successful rate and the best search accuracy among these algorithms.

4.4. *Comparisons on the algorithm reliability*

Table 5 also reveals that MDE offers a generally highest percentage of trials (reaching acceptable solutions) and the highest reliability averaged over all the testing functions. The MDE reaches the acceptable solutions with a successful ratio of 100% on twelve functions. For the mean reliability of all the testing functions, MDE offers the highest reliability of 80.63%, followed by CoDE, JADE, jDE, EPSDE and SaDE.

MDE outperforms most on both basic multimodal functions and rotated functions, owing to its mutation mechanism that avoids local optima. Further, such out

Table 6
Algorithms initialization

Algorithm	Reference
CLPSO	[32]
PSO with inertia weight (PSO-w)	[37]
PSO with constriction factor (PSO-cf)	[38]
Local version of PSO with inertia weight (PSO-w-local)	
Local version of PSO with constriction factor (PSO-cf-local)	[39]
UPSO	[40]
Fully informed particle swarm (FIPS)	[41]
FDR-PSO	[42]
CPSO-H	[43]

Table 7
Result comparisons among ten algorithms on testing functions

Function	**1**	**4**	**5**	**6**
PSO-w	9.78E-30(2.50E-29)	2.93E+1(2.51E+1)	3.94E-14(1.12E+0)	8.13E-3(7.16E-3)
PSO-cf	5.88E-100(5.40E-100)	1.11E+1(1.81E+0)	1.12E+0(8.65E-01)	2.06E-2(1.90E-2)
PSO-w-local	5.35E-100(4.41E-13)	2.39E+1(3.07E+0)	9.10E-8(8.11E-8)	5.91E-3(6.69E-3)
PSO-cf-local	7.70E-54(1.59E-53)	1.71E+1(9.16E-1)	5.33E-15(1.87E-15)	5.91E-3(8.70E-3)
UPSO	4.17E-87(3.15E-87)	1.51E+1(8.14E-1)	1.22E-15(3.16E-15)	1.66E-3(3.07E-3)
FDR	4.88E-102(1.53E-101)	**5.39E+0(1.76E+0)**	2.84E-14(4.10e-15)	1.01E-2(1.23E-2)
FIPS	2.69E-12(6.84E-13)	2.45 E+1(2.19 E-1)	4.81E-7(9.17E-8)	1.16E-6(1.87E-6)
CPSO-H	**1.16E-113(2.92E-113)**	7.08 E+0(8.01E+0)	4.93E-14(1.10E-14)	3.63E-2(3.60E-2)
CLPSO	4.46E-14(1.73E-14)	2.10 E+1(2.98E+0)	**0(0)**	3.14E-10(4.64E-10)
MDE	4.12E-77(4.23E-152)	2.46E+1(3.49E-1)	8.88E-16(0)	**0(0)**
Function	**7**	**8**	**9**	**11**
PSO-w	1.30E-4(3.30E-4)	2.90E+1(7.70E+0)	2.97E+1(1.39E+1)	1.71E+0(4.38E-1)
PSO-cf	4.10E+0(2.20E+0)	5.62E+1(9.76E+0)	2.85E+1(1.14E+1)	1.66E+000(1.10E+0)
PSO-w-local	4.94E-3(1.40E-2)	2.72E+1(7.58E+0)	2.08E+1(4.94E+0)	5.70E-1(7.60E-1)
PSO-cf-local	1.16E-1(2.79E-1)	4.53E+1(1.17E+1)	1.54E+1(1.67E+1)	1.78E-1(5.62E-1)
UPSO	9.60E+0(3.78E+0)	6.59E+1(1.22E+1)	6.34E+1(1.24E+1)	2.94E-1(6.71E-1)
FDR	7.49E-3(1.14E-2)	2.84E+1(8.71E+0)	1.44E+1(6.28E+0)	3.59E-01(5.93E-1)
FIPS	1.54E-1(1.48E-1)	7.30E+1(1.24E+1)	6.08E+1(8.35E+0)	**5.23E-7(1.42E-7)**
CPSO-H	7.82E-15(8.50E-15)	**0(0)**	1.00E-1(3.16E-1)	2.10E+0(3.84E-1)
CLPSO	3.45E-7(1.94E-7)	4.85E-10(3.63E-10)	4.36E-10(2.44E-10)	3.43E-04(1.91E-04)
MDE	**0(0)**	**0(0)**	**0(0)**	2.09E+1(4.3E-3)
Function	**12**	**13**	**14**	**15**
PSO-w	1.77E-2(1.53E-2)	7.00E+0(1.98E+0)	6.87E+1(2.05E+1)	6.32E+1(1.79E+1)
PSO-cf	8.62E-3(8.86E-3)	8.48E+000(2.54E+0)	7.13+1(1.66E+1)	7.88E+1(1.88E+1)
PSO-w-local	1.35E-2(1.12E-2)	5.96E+0(2.09E+0)	4.10E+1(7.93E+0)	5.67E+1(1.36E+1)
PSO-cf-local	1.30E-02(1.06E-2)	5.95E+000(2.95E+0)	4.66E+1(1.05E+1)	4.93E+1(1.11E+1)
UPSO	1.48E-02(3.12E-3)	1.85E+000(3.37E+0)	7.07E+1(1.70E+1)	7.74E+1(1.40E+1)
FDR	9.60E-03(1.24E-2)	2.50E+001(1.46E+0)	4.44E+001(1.37E+1)	4.36E+001(8.96E+0)
FIPS	6.92E-04(2.18E-3)	9.52E-02(9.53E-2)	7.41E+001(2.79E+1)	7.58E+001(1.92E+1)
CPSO-H	5.54E-02(3.97E-2)	1.43E+001(3.53E+0)	1.01E+002(2.21E+1)	8.80E+001(2.59E+1)
CLPSO	7.04E-10(1.25E-11)	3.07E+0(1.61E+0)	3.46E+1(4.59E+0)	3.77E+1(5.56E+0)
MDE	**0(0)**	**0(0)**	**0(0)**	**0(0)**

Table 8
Parameters setting

Group	k	d_c
1	0.3	1.5
2	0.4	2.0
3	0.5	2.5
4	[0.2 0.5]	[1.0 2.5]

performance has been achieved with the highest success rate on most testing functions.

4.5. Comparisons using t-tests

For a thorough comparison, the t-test has also been carried out [34, 35]. Table 4 presents the t values on every function of this two-tailed test with a significance level of 0.05 between the MDE and other DE algorithms. Rows "+ (Better)," "= (Same)," and "– (Worse)" give the number of functions that the MDE performs significantly better than, almost the same as, and significantly worse than the compared algorithm, respectively. Row "Score" shows the difference between the number of 1's and the number of –1's, which is used to give an overall comparison between the two algorithms.

For example, comparing MDE and jDE, the former significantly outperforms the latter on nine functions ($f_1, f_2, f_3, f_5, f_{10}, f_{12}, f_{13}, f_{14}, f_{15}$), does as same as the latter on seven functions ($f_4, f_6, f_7, f_8, f_9, f_{11}$, *and* f_{16}), and does worse on two functions (f_{17} and f_{18}), yielding a "Score" figure of merit of 9-2 = 7, indicating that the MDE generally outperforms the DE. Although it performs slightly weaker on some functions, the MDE in general offers much improved performance than all the DE algorithms compared, as confirmed in Table 5.

By above analysis, one may conclude that the MDE does not perform the best on the all testing functions. According to the "no free lunch" theorem [36], any elevated performance over one class of problems is offset by performance over another class. Therefore, we may

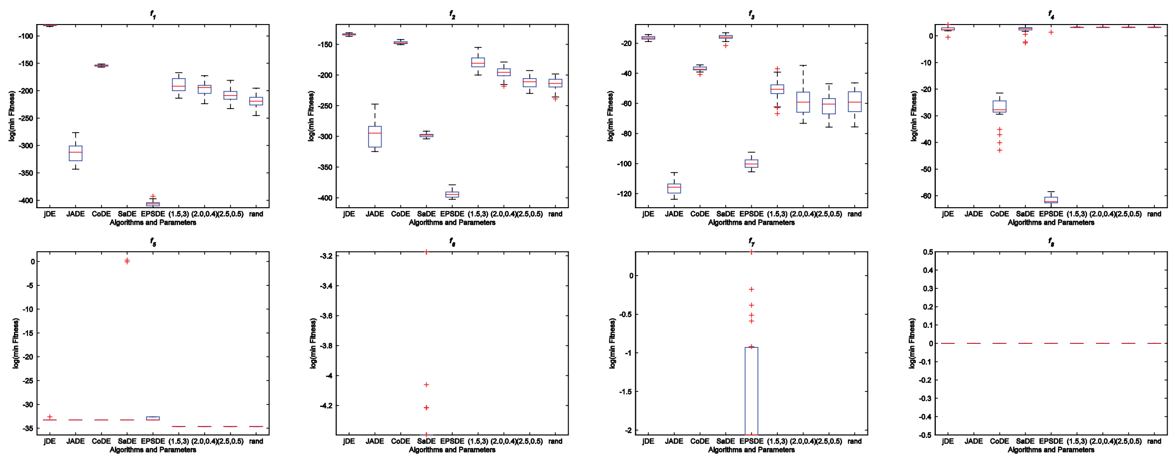

Fig. 2. MDE with four groups and other five algorithms on f_1 to f_8.

Table 9
Results comparisons among four groups on testing functions

	Group	1	2	3	4
f_1	Mean	1.29E-74	4.12E-77	1.12E-80	6.57E-87
	Std	3.39E-147	4.23E-152	1.85E-159	8.58E-172
f_2	Mean	1.95E-69	8.74E-80	9.07E-86	3.37E-88
	Std	9.51E-137	1.87E-157	1.91E-169	2.00E-174
f_3	Mean	3.72E-18	3.39E-17	1.58E-22	4.35E-22
	Std	2.79E-34	2.87E-32	6.19E-43	2.32E-42
f_4	Mean	2.43E+1	2.46E+1	2.48E+1	2.50E+1
	Std	1.48E+0	3.49E-1	6.79E-1	3.90E-1
f_5	Mean	8.88E-16	8.88E-16	8.88E-16	8.88E-16
	Std	0	0	0	0
f_6	Mean	0	0	0	0
	Std	0	0	0	0
f_7	Mean	6.65E-14	0	0	0
	Std	9.42E-26	0	0	0
f_8	Mean	0	0	0	0
	Std	0	0	0	0
f_9	Mean	0	0	0	0
	Std	0	0	0	0
f_{10}	Mean	6.99E-5	2.59E-4	1.89E-6	3.60E-5
	Std	1.06E-7	7.95E-7	3.52E-11	2.92E-8
f_{11}	Mean	2.09E+1	2.09E+1	2.09E+1	2.09E+1
	Std	2.9E-3	4.3E-3	2.3E-3	2.3E-3
f_{12}	Mean	0	0	0	0
	Std	0	0	0	0
f_{13}	Mean	4.33E-6	0	1.08E-14	0
	Std	4.69E-10	0	2.92E-27	0
f_{14}	Mean	0	0	0	0
	Std	0	0	0	0
f_{15}	Mean	0	0	0	3.48E-15
	Std	0	0	0	3.03E-28
f_{16}	Mean	7.73E-2	8.0E-1	1.79E-1	2.81E-1
	Std	1.54E-2	1.10E+1	6.63E-2	1.11E-1
f_{17}	Mean	6.96E-10	1.81E-9	4.00E-9	4.12E-9
	Std	3.88E-19	2.48E-18	4.12E-18	7.18E-18
f_{18}	Mean	1.66E-1	3.83E-1	6.58E-1	6.52E-1
	Std	2.71E-2	1.84E-1	2.46E-1	3.86E-1

not expect the best performance of MDE on all groups of functions.

4.6. *Comparison with nine PSO algorithms*

MDE is also compared with nine PSO algorithms presented in Table 6. Experiments are conducted to compare nine PSO algorithms on the above functions with 30 dimensions. The maximum FES is set at 200 000. All experiments are run 30 times. Table 7 summarizes the experimental result (results for PSO and related algorithms are taken from [32]). Overall, MDE scientifically outperforms the nine algorithms. In fact, CPSO-H, FDR and FIPS outperform MDE on one function respectively.

MDE is also compared with nine PSO algorithms presented in Table 6. Experiments are conducted to compare nine PSO algorithms on the above functions with 30 dimensions. The maximum FES is set at 200 000. All experiments are run 30 times. Table 7 summarizes the experimental result (results for PSO and related algorithms are taken from [37]). Overall, MDE scientifically outperforms the nine algorithms. In fact, CPSO-H, FDR and FIPS outperform MDE on one function respectively.

4.7. *Parameter sensitivity*

The parameters dc and K need to be optimized. The MDE algorithm run 25 times on each function with four different groups presented in Table 8.

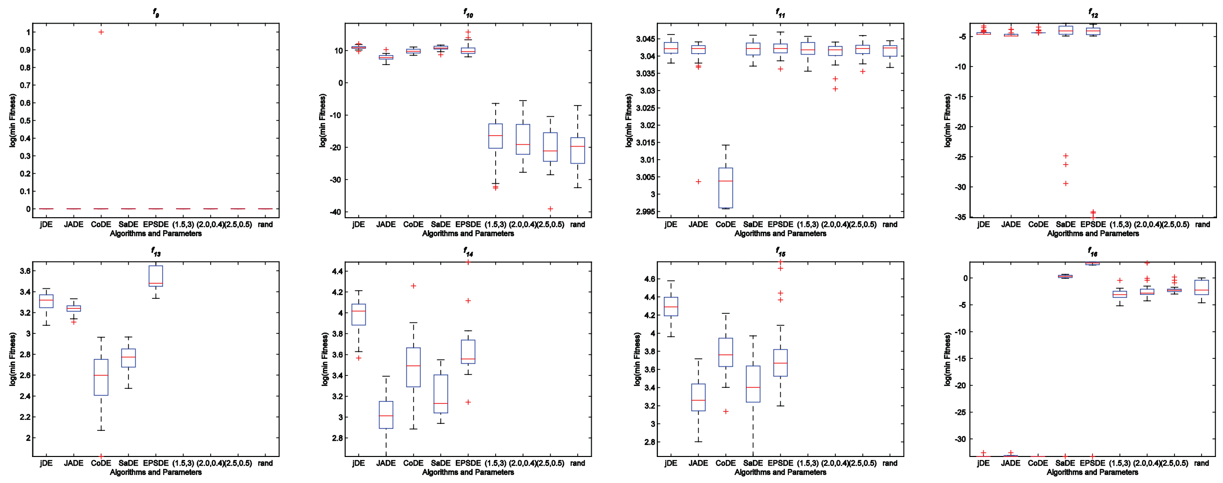

Fig. 3. MDE with four groups and other five algorithms on f_9 to f_{16}.

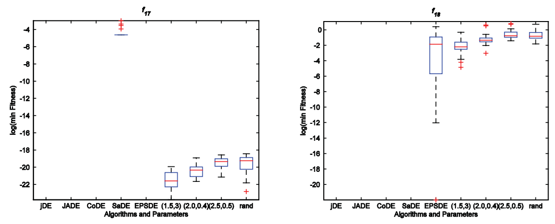

Fig. 4. MDE with four groups and other five algorithms on f_{17} to f_{18}.

The mean and standard deviation of the solutions in 25 independent runs are listed in Table 9. Figures 2–4 show the box plots of minimal values that MDE obtains with four different balanced parameters. The box has lines at the lower quartile, median, and upper quartile values. The whiskers are lines extending from each end of the box to show the extent of the remaining data. Outliers are data with values beyond the ends of the whiskers.

Functions f_7 and f_{13} are a little sensitive to the parameters. The rest functions results show any significant differences for most functions. Therefore, our MDE is not sensitive to the parameters. This is an advantage of our algorithm.

5. Conclusions

The performance of the DE algorithm is sensitive to the mutation strategy and control parameters. The adaptive method is proposed to determine the values of control parameters. Mutation operation based on DE/best/2 is proposed to improve generation strategy. Adaptive mutation is conducted to current population when the population clusters around local optima.

In comparison with five DE variants, MDE has been tested on a set of classic benchmark functions taken from the literature. It shows better or at least competitive optimization performance in terms of the accuracy, convergence rate and the reliability, compared to other adaptive DE algorithms. MDE also shows superiority over other PSO algorithms from the literature according to their reported results.

Acknowledgments

The authors would like to thank J. Brest, P. N. Suganthan, J. Zhang, R. Mallipeddi and Y. Wang for the code of jDE, SaDE, JADE, EPSDE and CoDE respectively.

This work was supported by the Priority Academic Program Development of Jiangsu Higher Education Institutions (PAPD), Social Science Foundation of Chinese Ministry of Education (nos.12YJC630271), Natural Science Fund for Colleges and Universities in Jiangsu Province (no.13KJB120008), China Natural Science Foundation (nos.71273139 and 71401078) and China Institute of Manufacturing Development (no. SK20130090-15).

References

[1] A.K. Qin, V.L. Huang and P.N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, *IEEE Transactions on Evolutionary Computation* **13** (2009), 398–417.

[2] M.G.H. Omran, A.P. Engelbrecht and A. Salman, Bare bones differential evolution, *European Journal of Operational Research* **196** (2009), 128–139.

[3] R. Storn and K. Price, Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces, *J Global Optimiz* **11** (1997), 341–359.

[4] Y. Tang, H. Gao and J. Kurths, Multiobjective identification of controlling areas in neuronal networks, *IEEE/ACM Transactions on Computational Biology and Bioinformatics* **2013**(10), 708-720.

[5] A. Ghosh, A. Mondal and S. Ghosh, Moving object detection using Markov Random Field and Distributed Differential Evolution, *Applied Soft Computing* **15** (2014), 121–135.

[6] Y. Tang, Z. Wang, H. Gao, S. Swift and J. Kurths, A constrained evolutionary computation method for detecting controlling regions of cortical networks, *IEEE/ACM Transactions on Computational Biology and Bioinformatics* **9** (2012), 1569–1581.

[7] R. Storn, Differential evolution design of an iir-filter, *IEEE International Conference on Evolutionary Computation IEEE* (1996), 268–273.

[8] K. Price, R. Storn and J. Lampinen, *Differential Evolution—A Practical Approach to Global Optimization.* Berlin, Germany: Springer-Verlag, 2005.

[9] J. Zhang and A.C. Sanderson, JADE: Adaptive differential evolution with optional external archive, *IEEE Trans Evolutionary Computation* **13** (2009), 945–958.

[10] K.V. Price, R.M. Storn and J.A. Lampinen, Differential evolution: A practical approach to global optimization, in: Natural Computing Series, Springer, Berlin, 2005.

[11] A. Iorio and X. Li, Solving rotated multi-objective optimization problems using differential evolution, in: *Australian Conference on Artificial Intelligence,* Cairns, Australia, 2004, pp. 861–872.

[12] Y. Wang, Z.X. Cai and Q.F. Zhang, Differential evolution with composite trial vector generation strategies and control parameters, *IEEE Transactions on Evolutionary Computation* **15** (2011), 55–66.

[13] R. Mallipeddi, P.N. Suganthan, Q.K. Pan and M.F. Tasgetiren, Differential evolution algorithm with ensemble of parameters and mutation strategies, *Applied Soft Computing* **11** (2011), 1679–1696.

[14] J. Liu and J. Lampinen, A fuzzy adaptive differential evolution algorithm, *Soft Comput* **9** (2005), 448–462.

[15] J. Brest, S. Greiner, B. Boskovic, M. Mernik and V. Zumer, Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems, *IEEE Trans Evolutionary Computation* **10** (2006), 646–657.

[16] Z. Yang, K. Tang and X. Yao, Self-adaptive differential evolution with neighborhood search, in *Proc IEEE Congr Evol Comput*, Hong Kong, China, 2008, pp. 1110–1116.

[17] J. Teo, Exploring dynamic self-adaptive populations in differential evolution, *Soft Comput: Fusion Found, Methodologies Applicat* **10** (2006), 673–686.

[18] H. Wang, Z.J. Wu and S. Rahnamayan, Enhanced opposition-based differential evolution for solving high-dimensional continuous optimization problems, *Soft Comput* **15** (2011), 2127–2140.

[19] H. Sharma, J.C. Bansal and K.V. Arya, Fitness based differential evolution, *Memetic Comp* **4** (2012), 303–316.

[20] M. Ali and M. Pant, Improving the performance of differential evolution algorithm using Cauchy mutation, *Soft Comput* **15** (2011), 991–1007.

[21] A.K. Qin and P.N. Suganthan, Self-adaptive differential evolution algorithm for numerical optimization, *The 2005 IEEE Congress on Evolutionary Computation*, 2005, pp. 1785–1791.

[22] R. Storn and K. Price, Differential evolution a simple and efficient heuristic for global optimization over continuous spaces, *J Global Optimization* **11** (1997), 341–359.

[23] K.V. Price, R.M. Storn and J.A. Lampinen, Differential Evolution: A Practical Approach to Global Optimization, 1st ed. New York: Springer-Verlag, 2005.

[24] B.V. Babu and M.M.L. Jehan, Differential evolution for multiobjective optimization, in *Proc IEEE Congr Evol Comput*, 2003, pp. 2696–2703.

[25] R. Gamperle, S.D. Muller and P. Koumoutsakos, A parameter study for differential evolution, in *Proc Advances Intell Syst, Fuzzy Syst, Evol Comput,* Crete, Greece, 2002, pp. 293–298.

[26] U. Pahner and K. Hameyer, Adaptive coupling of differential evolution and multiquadrics approximation for the tuning of the optimization process, *IEEE Trans Magnetics* **36** (2000), 1047–1051.

[27] E. Mezura-Montes, J. Velazquez-Reyes and C.A. Coello Coello, Modified differential evolution for constrained optimization, in *Proc IEEE Congr Evol Comput,* Vancouver, BC, 2006, pp. 25–32.

[28] X. Yao, Y. Liu and G.M. Lin, Evolutionary programming made faster, *IEEE Trans Evol Comput* **3** (1999), 82–102.

[29] C.Y. Lee and X. Yao, Evolutionary programming using mutations based on the levy probability distribution, *IEEE Trans Evol Comput* **8** (2004), 1–13.

[30] Z.G. Tu and L. Yong, A robust stochastic genetic algorithm (StGA) for global numerical optimization, *IEEE Trans Evol Comput* **8** (2004), 456–470.

[31] S.C. Esquivel and C.A. Coello Coello, On the use of particle swarm optimization with multimodal functions, in *Proc 2003 Congr Evol Comput*, vol. 2, Canberra, Australia, 2003, pp. 1130–1136.

[32] J.J. Liang, A.K. Qin, P.N. Suganthan and S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, *IEEE Transactions on Evolutionary Computation* **10** (2006), 281–294.

[33] Salomon, Reevaluating genetic algorithm performance under coordinate rotation of benchmark functions, *BioSystems* **39** (1996), 263–278.

[34] Z.H. Zhan, J. Zhang, Y. Li and H.S.H. Chung, Adaptive particle swarm optimization, *IEEE Transactions on Systems, Man, and Cybernetics B* **39** (2009), 1362–1381.

[35] Y.J. Zheng, H.F. Ling and Q. Guan, Adaptive parameters for a modified comprehensive learning particle swarm optimizer, *Mathematical Problems in Engineering* (2012). DOI 10.1155/2012/207318

[36] D.H. Wolpert and W.G. Macready, No free lunch theorems for optimization, *IEEE Trans Evol Comput* **1** (1997), 67–82.

[37] Y. Shi and R.C. Eberhart, A modified particle swarm optimizer, in *Proc IEEE Congr Evol Comput*, 1998, pp. 69–73.

[38] M. Clerc and J. Kennedy, The particle swarm-explosion, stability, and convergence in a multi dimensional complex space, *IEEE Trans Evol Comput* **6** (2002), 58–73.

[39] J. Kennedy and R. Mendes, Population structure and particle swarm performance, in *Proc IEEE Congr Evol Comput*, Honolulu, HI, 2002, pp. 1671–1676.

[40] K.E. Parsopoulos and M.N. Vrahatis, UPSO—A unified particle swarm optimization scheme, in *Lecture Series on Computational Sciences*, 2004, pp. 868–873.

[41] R. Mendes, J. Kennedy and J. Neves, The fully informed particle swarm: Simpler, maybe better, *IEEE Trans Evol Comput* **8** (2004), 204–210.

[42] T. Peram, K. Veeramachaneni and C.K. Mohan, Fitness-distance-ratio based particle swarm optimization, in *Proc Swarm Intelligence Symp*, 2003, pp. 174–181.

[43] F. van den Bergh and A.P. Engelbrecht, A cooperative approach to particle swarm optimization, *IEEE Trans Evol Comput* **8** (2004), 225–239.