Soft Comput (2011) 15:645–665
DOI 10.1007/s00500-010-0591-1

ORIGINAL PAPER

DE/BBO: a hybrid differential evolution with biogeography-based optimization for global numerical optimization

Wenyin Gong · Zhihua Cai · Charles X. Ling

Published online: 9 March 2010

Abstract Differential evolution (DE) is a fast and robust evolutionary algorithm for global optimization. It has been widely used in many areas. Biogeography-based optimization (BBO) is a new biogeography inspired algorithm. It mainly uses the biogeography-based migration operator to share the information among solutions. In this paper, we propose a hybrid DE with BBO, namely DE/BBO, for the global numerical optimization problem. DE/BBO combines the exploration of DE with the exploitation of BBO effectively, and hence it can generate the promising candidate solutions. To verify the performance of our proposed DE/BBO, 23 benchmark functions with a wide range of dimensions and diverse complexities are employed. Experimental results indicate that our approach is effective and efficient. Compared with other state-of-the-art DE approaches, DE/BBO performs better, or at least comparably, in terms of the quality of the final solutions and the convergence rate. In addition, the influence of the population size, dimensionality, different mutation schemes, and the self-adaptive control parameters of DE are also studied.

W. Gong (✉) · Z. Cai
School of Computer Science,
China University of Geosciences,
Wuhan, People's Republic of China
e-mail: cug11100304@yahoo.com.cn

Z. Cai
e-mail: zhcai@cug.edu.cn

C. X. Ling
Department of Computer Science,
The University of Western Ontario, London, Canada
e-mail: cling@csd.uwo.ca

Keywords Differential evolution · Biogeography-based optimization · Hybridization · Global numerical optimization · Exploration · Exploitation

1 Introduction

Evolutionary algorithms (EAs, including genetic algorithms, evolution strategies, evolutionary programming, and genetic programming) have received much attention regarding their potential as global optimization techniques (Bäck 1996), both in single and in multi-objective optimization. Inspired by the natural evolution and survival of the fittest, EAs utilize a collective learning process of a population of individuals. Descendants of individuals are generated using randomized operations such as mutation and recombination. Mutation corresponds to an self-replication of individuals, while recombination exchanges information between two or more existing individuals. According to a fitness measure, the selection process favors better individuals to reproduce more often than those that are relatively worse.

Differential evolution (DE) (Storn and Price 1997) is a simple yet powerful population-based, direct search algorithm with the generation-and-test feature for global optimization problems using real-valued parameters. DE uses the distance and direction information from the current population to guide the further search. It won the third place at the first International Contest on Evolutionary Computation on a real-valued function test-suite (Storn and Price 2008). Among DE's advantages are its simple structure, ease of use, speed and robustness. Price and Storn (1997) gave the working principle of DE with single scheme. Later on, they suggested ten different schemes of DE (Storn and Price 2008; Price et al. 2005). However, DE

has been shown to have certain weaknesses, especially if the global optimum should be located using a limited number of fitness function evaluations (NFFEs). In addition, DE is good at exploring the search space and locating the region of global minimum, but it is slow exploiting of the solution (Noman and Iba 2008).

Biogeography-based optimization (BBO), proposed by Simon (2008a, b), is a new global optimization algorithm based on the biogeography theory, which is the study of the geographical distribution of biological organisms. Similar to genetic algorithms (GAs), BBO is a population-based, stochastic global optimizer. In the original BBO algorithm, each solution of the population is a vector of integers. BBO adopts the migration operator to share information between solutions. This feature is similar to other biology-based algorithms, such as GAs and PSO. It makes BBO applicable to many of the same types of problems that GAs and PSO are used for. However, BBO also has several unique features compared with biology-based algorithms. For example, it maintains its set of solutions from one iteration to the next one (Simon 2008a, b). Simon compared BBO with seven state-of-the-art EAs over 14 benchmark functions and a real-world sensor selection problem. The results demonstrated the good performance of BBO. With the migration operator, BBO has a good exploitation ability.

Hybridization of EAs is getting more and more popular due to their capabilities in handling several real world problems (Grosan et al. 2009). A good review of hybrid metaheuristics with EAs specializing in intensification and diversification can be found in Lozano and García-Martínez (2010). In order to balance the exploration and the exploitation of DE, in this paper, we propose a hybrid DE with BBO, referred to as DE/BBO, for the global numerical optimization problems. In DE/BBO, a hybrid migration operator is proposed, which combines the exploration of DE with the exploitation of BBO effectively. Experiments have been tested on 23 benchmark functions chosen from the literature. In addition, five performance criteria are employed to fairly compare our approach with other algorithms. Furthermore, the influence of the population size, dimensionality, different mutation schemes, and the self-adaptive control parameters of DE are also investigated.

The rest of this paper is organized as follows. Section 2 describes briefly function optimization problem, the DE algorithm, and the BBO algorithm. In Sect. 3, some related work of DE are presented. Our proposed approach is presented in detail in Sect. 4. In Sect. 5, we verify our approach through 23 benchmark functions. Moreover, the experimental results are compared with several other approaches. The last section, Sect. 6, is devoted to conclusions and future work.

2 Preliminary

2.1 Problem definition

Global numerical optimization problems are frequently arisen in almost every field of engineering design, applied sciences, molecular biology and other scientific applications. Without loss of generality, the global minimization problem can be formalized as a pair (S, f), where $S \subseteq R^D$ is a bounded set on R^D and $f : S \rightarrow R$ is a D-dimensional real-valued function. The problem is to find a point $X^* \in S$ such that $f(X^*)$ is the global minimum on S (Yao et al. 1999). More specifically, it is required to find an $X^* \in S$ such that

$$\forall X \in S : f(X^*) \leq f(X) \tag{1}$$

where f does not need to be continuous but it must be bounded. In this work, we only consider the unconstrained function optimization.

In global numerical optimization problems, the major challenge is that an algorithm may be trapped in the local optima of the objective function. This issue is particularly challenging when the dimension is high. Recently, using the evolutionary computation (EC) (Bäck 1996) to solve the global optimization has been very active, producing different kinds of EC for optimization in the continuous domain, such as GAs (Zhong et al. 2004; Herrera et al. 1998; Herrera 2000), evolution strategy (Yao and Liu 1997; Hansen and Kern 2004; Auger and Hansen 2004), evolutionary programming (Yao et al. 1999), particle swarm optimization (PSO) (Liang et al. 2006; Langdon and Poli 2007), immune clonal algorithm (Jiao et al. 2008), DE (Storn and Price 1997), memetic algorithms (Lozano et al. 2004), etc.

Algorithm 1 The DE algorithm with DE/rand/1/bin scheme

```
Generate the initial population P
Evaluate the fitness for each individual in P
while The halting criterion is not satisfied do
    for i = 1 to NP do
        Select uniform randomly r1 ≠ r2 ≠ r3 ≠ i
        j_rand = rndint(1, D)
        for j = 1 to D do
            if rndreal_j[0, 1) < CR or j == j_rand then
                U_i(j) = X_r1(j) + F × (X_r2(j) − X_r3(j))
            else
                U_i(j) = X_i(j)
            end if
        end for
    end for
    for i = 1 to NP do
        Evaluate the offspring U_i
        if U_i is better than P_i then
            P_i = U_i
        end if
    end for
end while
```

2.2 Differential evolution

Differential evolution (Storn and Price 1997) is a simple EA that creates new candidate solutions by combining the parent individual and several other individuals of the same population. A candidate replaces the parent only if it has better fitness. Among DE's advantages are its simple structure, ease of use, speed and robustness. Due to these advantages, it has been applied to many real-word applications, such as data mining (Alatas et al. 2008; Das et al. 2008), pattern recognition, digital filter design, neural network training, etc. (Price et al. 2005; Feoktistov 2006; Chakraborty 2008). Most recently, DE has also been used for the global permutation-based combinatorial optimization problems (Onwubolu and Davendra 2009).

The pseudo-code of the original DE algorithm is shown in Algorithm 1. Where D is the number of decision variables. NP is the size of the parent population P. F is the mutation scaling factor. CR is the probability of crossover operator. $X_i(j)$ is the jth variable of the solution X_i. U_i is the offspring. $\text{rndint}(1, D)$ is a uniformly distributed random integer number between 1 and n. And $\text{rndreal}_j[0, 1)$ is a uniformly distributed random real number in [0, 1). Many schemes of creation of a candidate are possible. We use the DE/rand/1/bin scheme (see lines 6–13 of Algorithm 1) described in Algorithm 1 (more details on DE/rand/1/bin and other DE schemes can be found in Storn and Price (2008) and Price et al. (2005).

From Algorithm 1, we can see that there are only three control parameters in this algorithm. These are NP, F and CR. As for the terminal conditions, one can either fix the maximum NFFEs Max_NFFEs or the precision of a desired solution VTR (value to reach).

Algorithm 2 Habitat migration

```
1: for i = 1 to NP do
2:     Select X_i with probability ∝ λ_i
3:     if rndreal(0, 1) < λ_i then
4:         for j = 1 to NP do
5:             Select X_j with probability ∝ μ_j
6:             if rndreal(0, 1) < μ_j then
7:                 Randomly select a variable σ from X_j
8:                 Replace the corresponding variable in X_i with σ
9:             end if
10:        end for
11:    end if
12: end for
```

2.3 Biogeography-based optimization

BBO (Simon 2008a, b) is a new population-based, biogeography inspired global optimization algorithm. In BBO, each individual is considered as a "habitat" with a habitat suitability index (HSI), which is similar to the fitness of EAs, to measure the individual. A good solution is analogous to an island with a high HSI, and a poor solution indicates an island with a low HSI. High HSI solutions tend to share their features with low HSI solutions. Low HSI solutions accept a lot of new features from high HSI solutions.

In BBO, each individual has its own immigration rate λ and emigration rate μ. A good solution has higher μ and lower λ, vice versa. The immigration rate and the emigration rate are functions of the number of species in the habitat. They can be calculated as follows:

$$\lambda_k = I\left(1 - \frac{k}{n}\right) \tag{2}$$

$$\mu_k = E\left(\frac{k}{n}\right) \tag{3}$$

where I is the maximum possible immigration rate. E is the maximum possible emigration rate. k is the number of species of the kth individual in the ordered population according to the fitness. For example, for the minimization problem the population is ordered with the descent order based on the fitness, so that for the best solution $k = n$, and $\mu_k = E$, $\lambda_k = 0$, it means that this solution can share more information to other poor solutions. n is the maximum number of species. Note that Eqs. 2 and 3 are just one method for calculating λ and μ. There are other different options to assign them based on different specie models (Simon 2008a, b). In this work, we use Eqs. 2 and 3 to calculate the immigration rate and the emigration rate as proposed in Simon (2008a, b).

Suppose that we have a global optimization problem and a population of candidate individuals. The individual is represented by a D-dimensional vector. The population consists of $NP = n$ parameter vectors. In BBO, there are two main operators, the migration and the mutation. One option for implementing the migration operator can be described in Algorithm 2.[1] Where $\text{rndreal}(0, 1)$ is a uniformly distributed random real number in (0,1) and $X_i(j)$ is the jth SIV of the solution X_i. With the migration operator, BBO can share the information among solutions. Especially, poor solutions tend to accept more useful information from good solutions. This makes BBO be good at exploiting the information of the current population. More details about the two operators can be found in Simon (2008a, b) and in the Matlab code (Simon 2008a, b).

[1] Since the mutation operator of BBO is not used in our approach, we do not describe it here. Interested readers can refer to (Simon 2008a, b).

3 Related work to DE

Some previous researches pointed out that there are three main drawbacks of the original DE algorithm. First, the parameters of DE are problem dependent and the choice of them is often critical for the performance of DE (Gäperle et al. 2002; Liu and Lampinen 2005). Second, choosing the best among different mutation schemes available for DE is also not easy for a specific problem (Qin and Suganthan 2005; Qin et al. 2009). Third, DE is good at exploring the search space and locating the region of global minimum, but it is slow exploiting of the solution (Noman and Iba 2008). Due to these drawbacks, many researchers are now working on the improvement of DE, and many variants are presented.

Adapting the DE's control parameters is one possible improvement. Liu and Lampinen (2005) proposed a Fuzzy Adaptive DE (FADE), which employs fuzzy logic controllers to adapt the mutation and crossover control parameters. Brest et al. (2006) proposed self-adapting control parameter settings. Their proposed approach encodes the F and CR parameters into the chromosome and uses a self-adaptive control mechanism to change them. Salman et al. (2007) proposed a self-adaptive DE (SDE) algorithm that eliminates the need for manual tuning of control parameters. In SDE, the mutation scaling factor F is self-adapted by a mutation strategy similar to the mutation operator of DE. Nobakhti and Wang (2008) proposed a randomized adaptive differential evolution (RADE) method, where a simple randomized self-adaptive scheme was proposed for the mutation weighting factor F. Das et al. (2005) proposed two variants of DE, DERSF and DETVSF, that use varying scale factors. They concluded that those variants outperform the original DE. Teo (2006) presented a dynamic self-adaptive populations DE, where the population size is self-adapting. Through five De Jong's test functions, they showed that DE with self-adaptive populations produced highly competitive results. Brest and Maučec (2008) proposed an improved DE method, where the population size is gradually reduced. They concluded that their approach improved efficiency and robustness of DE. Teng et al. (2009) proposed a variant of DE where two different techniques (absolute encoding and relative encoding) were used to implement the self-adaptive population size. The authors concluded that DE with the self-adaptive population size using relative encoding performed well (Teng et al. 2009).

Qin and Suganthan (2005) proposed a self-adaptive DE algorithm. The aim of their work was to allow DE to switch between two schemes: "DE/rand/1/bin" and "DE/best/2/bin" and also to adapt the F and CR values. The approach performed well on several benchmark problems. Recently, Qin et al. (2009) extended their previous work (Qin and Suganthan 2005). In their SaDE, four schemes were adopted. And different CR values were also used for different mutation schemes. Their proposed algorithm outperformed the original DE and some other compared adaptive/self-adaptive DE variants (Qin et al. 2009).

Hybridization with other different algorithms is another direction for the improvement of DE. Fan and Lampinen (2003) proposed a new version of DE that uses an additional mutation operation called trigonometric mutation operation. They showed that the modified DE algorithm can outperform the classic DE algorithm for some benchmarks and real-world problems. Sun et al. (2005) proposed a new hybrid algorithm based on a combination of DE with estimation of distribution algorithm (EDA). This technique uses a probability model to determine promising regions in order to focus the search process on those areas. Gong et al. (2006) employed the two level orthogonal crossover to improve the performance of DE. They showed that the proposed approach performs better than the classical DE in terms of the quality, speed, and stability of the final solutions. Noman and Iba (2005) proposed fittest individual refinement, a crossover-based local search (LS) method DE to solve the high dimensional problems. Based on their previous work (Noman and Iba 2005), they incorporated LS into the classical DE in Noman and Iba (2008). They presented a LS technique to solve this problem by adaptively adjusting the length of the search, using a hill-climbing heuristic. Through the experiments, they showed that the proposed new version of DE performs better, or at least comparably, to classic DE algorithm. Kaelo and Ali (2007) adopted the attraction-repulsion concept of electromagnetism-like algorithm to boost the mutation operation of the original DE. Wang et al. (2007) proposed a dynamic clustering-based DE for global optimization, where a hierarchical clustering method is dynamically incorporated in DE. Experiments on 28 benchmark problems, including 13 high dimensional functions, showed that the new method is able to find near optimal solutions efficiently (Wang et al. 2007). Rahnamayan et al. (2008) proposed a novel initialization approach which employs opposition-based learning to generate initial population. Through a comprehensive set of benchmark functions they showed that replacing the random initialization with the opposition-based population initialization in DE can accelerate convergence speed. In Caponio et al. (2009), proposed a memetic DE method, namely SFMDE. In SFMDE, the PSO algorithm is used in the beginning of the evolutionary process to generate the super-fit solutions. Additionally, two local searchers are adaptively used to refine the individuals of the population.

4 Our approach: DE/BBO

As mentioned above, DE is good at exploring the search space and locating the region of global minimum. However, it is slow exploiting of the solution (Noman and Iba 2008). On the other hand, BBO has a good exploitation for global optimization (Simon 2008a, b). Based on these considerations, in order to balance the exploration and the exploitation of DE, in this work, we propose a hybrid DE approach, called DE/BBO, which combines the exploration of DE with the exploitation of BBO effectively. Our proposed DE/BBO approach is described as follows.

4.1 Hybrid migration operator

The main operator of DE/BBO is the hybrid migration operator, which hybridizes the DE operator with the migration operator of BBO, described in Algorithm 3. From Algorithm 3 we can see that the offspring U_i is possibly constituted by three components: the DE mutant, the migration of other solutions, and its corresponding parent X_i. The core idea of the proposed hybrid migration operator is based on two considerations. First, good solutions would be less destroyed, while poor solutions can accept a lot of new features from good solutions. In this sense, the current population can be exploited sufficiently. Second, the mutation operator of DE is able to explore the new search space and make the algorithm more robust. According to Lozano and García-Martínez (2010) the original migration operator in DE/BBO focuses on the intensification; while, the DE mutation operator emphasizes on the diversification. From the analysis, it can be seen that the hybrid migration operator can balance the exploration and the exploitation effectively. It is worth pointing out that in Algorithm 1 the "DE/rand/1" mutation operator is illustrated, however, other mutation operators of DE can also be used in our proposed hybrid migration operator. The influence of different mutation operators will be discussed in Sect. 5.6.

Algorithm 3 Hybrid migration operator of DE/BBO

for $i = 1$ to NP **do**
  Select uniform randomly $r_1 \neq r_2 \neq r_3 \neq i$
  j_{rand} = rndint$(1, D)$
  for $j = 1$ to D **do**
    if rndreal$(0, 1) < \lambda_i$ **then**
      if rndreal$_j[0, 1) < CR$ or $j == j_{rand}$ **then**
        $U_i(j) = X_{r_1}(j) + F \times \left(X_{r_2}(j) - X_{r_3}(j)\right)$ {*The original mutation operator of DE.*}
      else
        Select X_k with probability $\propto \mu_k$
        $U_i(j) = X_k(j)$
      end if
    else
      $U_i(j) = X_i(j)$
    end if
  end for
end for

4.2 Boundary constraints

In order to keep the solution of bound-constrained problems feasible, those trial parameters that violate boundary constraints should be reflected back from the bound by the amount of violation. In this work, the following repair rule is applied

$$X(i) = \begin{cases} l_i + \text{rndreal}_i[0, 1] \times (u_i - l_i) & \text{if } X(i) < l_i \\ u_i - \text{rndreal}_i[0, 1] \times (u_i - l_i) & \text{if } X(i) > u_i \end{cases} \quad (4)$$

where rndreal$_i[0, 1]$ is the uniform random variable from [0,1] in each dimension i.

4.3 Main procedure of DE/BBO

By incorporating the above-mentioned hybrid migration operator into DE, the DE/BBO approach is developed and shown in Algorithm 4. Compared with the original DE algorithm described in Algorithm 1, our approach needs only a small extra computational cost in sorting the population and calculating the migration rates. In addition, the structure of our proposed DE/BBO is also very simple. Moreover, DE/BBO is able to explore the new search space with the mutation operator of DE and to exploit the population information with the migration operator of BBO. This feature overcomes the lack of exploitation of the original DE algorithm. Note that the only difference between DE/BBO and the original DE algorithm is that the hybrid migration operator is used to replace the original DE mutation operator. In DE/BBO, the selection method of DE is kept the same; therefore, DE/BBO is also an elitist method like the original DE algorithm.

Algorithm 4 The main procedure of DE/BBO

Generate the initial population P
Evaluate the fitness for each individual in P
while The halting criterion is not satisfied **do**
  Sort the population from worst to best
  For each individual, map the fitness to the number of species
  Calculate the immigration rate λ_i and the emigration rate μ_i for each individual X_i
  Modify the population with the hybrid migration operator shown in Algorithm 3
  for $i = 1$ to NP **do**
    Evaluate the offspring U_i
    if U_i is better than P_i **then**
      $P_i = U_i$
    end if
  end for
end while

5 Experimental results

In order to verity the performance of DE/BBO, 23 benchmark functions are chosen from Yao et al. (1999) Since we do not make any modification of these functions, they are only briefly described in Table 1. A more detailed description of these functions can be found in Yao et al. (1999), where the functions were divided into three categories: unimodal functions, multimodal functions with many local minima, and multimodal functions with a few local minima.

Functions f01–f13 are high-dimensional and scalable problems. Functions f01–f05 are unimodal. Function f06 is the step function, which has one minimum and is discontinuous. Function f07 is a noisy quartic function, where *random* [0,1) is a uniformly distributed random variable in [0,1). Functions f08–f13 are multimodal functions where the number of local minima increases exponentially with the problem dimension. They appear to be the most difficult class of problems for many optimization algorithms. Functions f14–f23 are low-dimensional functions that have only a few local minima.

5.1 Experimental setup

For DE/BBO, we have chosen a reasonable set of value and have not made any effort in finding the best parameter settings. For all experiments, we use the parameters shown in Table 2 unless a change is mentioned. The maximum NFFEs (Max_NFFEs) for each function at $D = 30$ are listed in the second column of Table 3.

Moreover, in our experiments, each function is optimized over 50 independent runs. We also use the same set of initial random populations to evaluate different algorithms in a similar way done in Noman and Iba (2008). All the algorithms are implemented in standard C++. The source code can be obtained from the first author upon request.

5.2 Performance criteria

Five performance criteria are selected from the literature (Rahnamayan et al. 2008; Suganthan et al. 2005) to evaluate the performance of the algorithms. These criteria are described as follows.

Table 1 Benchmark functions used in our experimental studies

Functions	Name	D	S	Optimal
f01	Sphere model	30	$[-100, 100]^D$	0
f02	Schwefel's problem 2.22	30	$[-10, 10]^D$	0
f03	Schwefel's problem 1.2	30	$[-100, 100]^D$	0
f04	Schwefel's problem 2.21	30	$[-100, 100]^D$	0
f05	Generalized Rosenbrock's functions	30	$[-30, 30]^D$	0
f06	Step function	30	$[-100, 100]^D$	0
f07	Quartic function	30	$[-1.28, 1.28]^D$	0
f08	Generalized Schwefel's problem 2.26	30	$[-500, 500]^D$	-12569.5
f09	Generalized Rastrigin's function	30	$[-5.12, 5.12]^D$	0
f10	Ackley's function	30	$[-32, 32]^D$	0
f11	Generalized Griewank function	30	$[-600, 600]^D$	0
f12	Generalized Penalized function 1	30	$[-50, 50]^D$	0
f13	Generalized Penalized function 2	30	$[-50, 50]^D$	0
f14	Shekel's Foxholes function	2	$[-65.536, 65.536]^D$	1
f15	Kowalik's function	4	$[-5, 5]^D$	0.003075
f16	Six-Hump Camel-Back function	2	$[-5, 5]^D$	-1.0316285
f17	Branin Function	2	$[-5, 10] \times [0, 15]$	0.398
f18	Glodstein-Price function	2	$[0, 1]^D$	3
f19	Hartman's function 1	3	$[0, 1]^D$	-3.86
f20	Hartman's function 2	6	$[0, 1]^D$	-3.32
f21	Shekel's Function 1	4	$[0, 10]^D$	-10.1532
f22	Shekel's function 2	4	$[0, 10]^D$	-10.4029
f23	Shekel's function 3	4	$[0, 10]^D$	-10.5364

More details of all functions can be found in Yao et al. (1999)

Table 2 The parameter settings of the DE/BBO method used in this work

Parameter	Value
Population size: *NP*	100 (Noman and Iba 2008; Yao et al. 1999; Brest et al. 2006; Rahnamayan et al. 2008)
Habitat modification probability	1.0 (Simon 2008a, b)
Maximum immigration rate: *I*	1.0 (Simon 2008a, b)
Maximum emigration rate: *E*	1.0 (Simon 2008a, b)
Scaling factor: *F*	rndreal$(0.1, 1.0)$ (Price et al. 2005; Chakraborty 2008)
Crossover probability: *CR*	0.9 (Storn and Price 1997; Liu and Lampinen 2005; Teo 2006; Rahnamayan et al. 2008)
DE mutation scheme	DE/rand/1/bin (Storn and Price 1997; Noman and Iba 2008; Liu and Lampinen 2005; Teo 2006)
Value to reach: VTR	10^{-8} (Suganthan et al. 2005), except for f07 of VTR = 10^{-2}

Table 3 Best error values of DE/BBO, DE, and BBO on all test functions, where "Mean" indicates the mean best error values found in the last generation, "Std Dev" stands for the standard deviation

F	Max_NFFEs	DE/BBO			DE			BBO			1 vs. 2	1 vs. 3
		Mean	Std Dev	SR	Mean	Std Dev	SR	Mean	Std Dev	SR	*t test*	*t test*
f01	150,000	**8.66E−28**	5.21E−28	50	1.10E−19	1.34E−19	50	8.86E−01	3.26E−01	0	−5.81†	−19.21†
f02	200,000	**0.00E+00**	0.00E+00	50	1.66E−15	8.87E−16	50	2.42E−01	4.58E−02	0	−13.24†	−37.36†
f03	500,000	2.26E−03	1.58E−03	0	**8.19E−12**	1.65E−11	50	4.16E+02	2.02E+02	0	10.10*	−14.58†
f04	500,000	**1.89E−15**	8.85E−16	50	7.83E+00	3.78E+00	0	7.76E−01	1.72E−01	0	−14.65†	−31.96†
f05	500,000	1.90E+01	7.52E+00	0	**8.41E−01**	1.53E+00	6	9.14E+01	3.78E+01	0	16.73*	−13.28†
f06	150,000	**0.00E+00**	0.00E+00	50	**0.00E+00**	0.00E+00	50	2.80E−01	5.36E−01	38	0	−3.69†
f07	300,000	**3.44E−03**	8.27E−04	50	3.49E−03	9.60E−04	50	1.90E−02	7.29E−03	4	−0.29	−14.96†
f08	300,000	**0.00E+00**	0.00E+00	50	4.28E+02	4.69E+02	1	5.09E−01	1.65E−01	0	−6.45†	−21.78†
f09	300,000	**0.00E+00**	0.00E+00	50	1.14E+01	7.57E+00	0	8.50E−02	3.42E−02	0	−10.61†	−17.61†
f10	150,000	**1.07E−14**	1.90E−15	50	6.73E−11	2.86E−11	50	3.48E−01	7.06E−02	0	−16.66†	−34.81†
f11	200,000	**0.00E+00**	0.00E+00	50	1.23E−03	3.16E−03	43	4.82E−01	1.27E−01	0	−2.76†	−26.93†
f12	150,000	**7.16E−29**	6.30E−29	50	2.07E−03	1.47E−02	49	5.29E−03	5.21E−03	0	−1.00	−7.18†
f13	150,000	**9.81E−27**	7.10E−27	50	7.19E−02	5.09E−01	49	1.42E−01	5.14E−02	0	−1.00	−19.50†
f14	10,000	**0.00E+00**	0.00E+00	50	2.75E−13	1.55E−12	50	8.85E−06	2.74E−05	14	−1.26	−2.28†
f15	40,000	3.84E−12	2.70E−11	50	**4.94E−19**	5.20E−19	50	5.92E−04	2.68E−04	0	1.01	−15.65†
f16	10,000	1.15E−12	6.39E−12	50	**1.98E−13**	4.12E−13	50	6.75E−04	1.09E−03	0	1.05	−4.37†
f17	10,000	**2.92E−10**	1.38E−09	50	7.32E−10	2.21E−09	49	4.39E−04	4.26E−04	0	−1.19	−7.30†
f18	10,000	9.15E−13	6.51E−15	50	9.14E−13	5.01E−15	50	7.86E−03	9.57E−03	0	0.70	−5.81†
f19	10,000	**0.00E+00**	0.00E+00	50	1.36E−14	6.40E−15	50	2.51E−04	2.62E−04	0	−15.05†	−6.76†
f20	20,000	**0.00E+00**	0.00E+00	50	4.76E−03	2.35E−02	47	1.46E−02	3.90E−02	0	−1.43	−2.64†
f21	10,000	3.59E−03	1.44E−02	15	**6.83E−06**	1.26E−05	0	5.18E+00	3.34E+00	0	1.76	−10.95†
f22	10,000	**3.14E−07**	1.36E−06	29	7.26E−06	4.53E−05	1	3.67E+00	3.40E+00	0	−1.08	−7.63†
f23	10,000	**2.50E−08**	5.37E−08	27	3.70E−06	1.75E−05	0	2.73E+00	3.29E+00	0	−1.49	−5.87†

"1 vs. 2" means "DE/BBO vs. DE" and "1 vs. 3" means "DE/BBO vs. BBO". Hereafter, a result with boldface means better value found

†,* The value of t with 49 degrees of freedom is significant at $\alpha = 0.05$ by two-tailed test

* means that the corresponding algorithm is better than our proposed DE/BBO method

- *Error:* Suganthan et al. (2005): The error of a solution X is defined as $f(X) - f(X^*)$, where X^* is the global optimum of the function. The minimum error is recorded when the Max_NFFEs is reached in 50 runs. Also the average and standard deviation of the error values are calculated.
- *NFFEs:* Suganthan et al. (2005). The number of fitness function evaluations (NFFEs) is also recorded when the VTR is reached. The average and standard deviation of the NFFEs values are calculated.
- *Number of successful runs (SR):* Suganthan et al. (2005): The number of successful runs is recorded

Table 4 NFFEs required to obtain accuracy levels less than VTR

F	DE/BBO			DE			BBO			1 vs. 2	1 vs. 3
	Mean	Std Dev	SR	Mean	Std Dev	SR	Mean	Std Dev	SR	*AR*	*AR*
f01	**59,926**	745.5	50	79,688	1,858.8	50	NA	NA	0	1.33	NA
f02	**82,004**	983.9	50	119,764	1,871.2	50	NA	NA	0	1.46	NA
f03	NA	NA	0	**385,990**	17,193.4	50	NA	NA	0	NA	NA
f04	**296,572**	4,969.9	50	NA	NA	0	NA	NA	0	NA	NA
f05	NA	NA	0	**448,583**	60,428.8	6	NA	NA	0	NA	NA
f06	**21,590**	573.3	50	28,874	2,014.5	50	119,042	16,882.8	38	1.34	5.51
f07	109,574	21,005.8	50	**103,136**	29,677.7	50	205,000	23,896.2	4	0.94	1.87
f08	**95,952**	3,126.7	50	251,700	0	1	NA	NA	0	2.62	NA
f09	**170,226**	8,379.0	50	NA	NA	0	NA	NA	0	NA	NA
f10	**91,308**	922.7	50	122,340	2,179.6	50	NA	NA	0	1.34	NA
f11	**62,042**	1219.6	50	81,986	1,795.8	43	NA	NA	0	1.32	NA
f12	**54,482**	873.3	50	71,183	4,700.9	49	NA	NA	0	1.31	NA
f13	**64,772**	1,133.4	50	93,298	16,916.8	49	NA	NA	0	1.44	NA
f14	**4,532**	719.5	50	6,768	766.0	50	5,757	2,351.3	14	1.49	1.27
f15	24,028	3,279.3	50	**12,590**	977.8	50	NA	NA	0	0.52	NA
f16	**5,676**	1,012.7	50	5,760	632.5	50	NA	NA	0	1.01	NA
f17	**7,138**	1,404.9	50	7,271	1,098.7	49	NA	NA	0	1.02	NA
f18	5,050	374.3	50	**4,610**	292.9	50	NA	NA	0	0.91	NA
f19	**4,808**	352.2	50	5,434	346.8	50	NA	NA	0	1.13	NA
f20	**9,614**	705.1	50	14,161	1,553.3	47	NA	NA	0	1.47	NA
f21	**9,560**	397.9	15	NA	NA	0	NA	NA	0	NA	NA
f22	**9,527**	349.4	29	10,000	0	1	NA	NA	0	1.05	NA
f23	**9,533**	362.7	27	NA	NA	0	NA	NA	0	NA	NA

"NA" indicates the accuracy level is not obtained after Max_NFFEs. "1 vs. 2" means "DE/BBO vs. DE" and "1 vs. 3" means "DE/BBO vs. BBO"

when the VTR is reached before the Max_NFFEs condition terminates the trial.

- *Convergence graphs:* Suganthan et al. (2005): The convergence graphs show the mean error performance of the best solution over the total runs, in the respective experiments.
- *Acceleration rate (AR):* Rahnamayan et al. (2008): This criterion is used to compare the convergence speeds between DE/BBO and other algorithms. It is defined as follows:

$$\mathrm{AR} = \frac{\mathrm{NFFEs}_{\mathrm{other}}}{\mathrm{NFFEs}_{\mathrm{DE/BBO}}} \tag{5}$$

where AR $>$ 1 indicates DE/BBO converges faster than its competitor.

5.3 General performance of DE/BBO

In order to show the superiority of our proposed DE/BBO approach, we compare it with the original DE algorithm and the BBO algorithm. The parameters used for DE/BBO and DE are the same as described in Sect. 5.1. The parameters of BBO are set as in Simon 2008a, b), and the mutation operator with $m_{\max} = 0.005$ is also used in our experiments. All functions are tested for 50 independent runs. Table 3 shows the best error values of DE/BBO, DE, and BBO on all test functions. The average and standard deviation of NFFEs are shown in Table 4. In addition, some representative convergence graphs of DE/BBO, DE, and BBO are shown in Fig. 1.

5.3.1 When compared with DE

From Table 3 we can see that DE/BBO is significantly better than DE on eight functions. However, DE/BBO is outperformed by DE on two functions (f03 and f05). For the rest 13 functions, there are no significant difference based on the *t* test.[2] For the multimodal functions with many local minima (f08–f13), DE/BBO can obtain the VTR $= 10^{-8}$ over all 50 runs within the Max_NFFEs.

[2] The paired *t*-test determines whether two paired sets differ from each other in a significant way under the assumptions that the paired differences are independent and identically normally distributed (Goulden 1956).

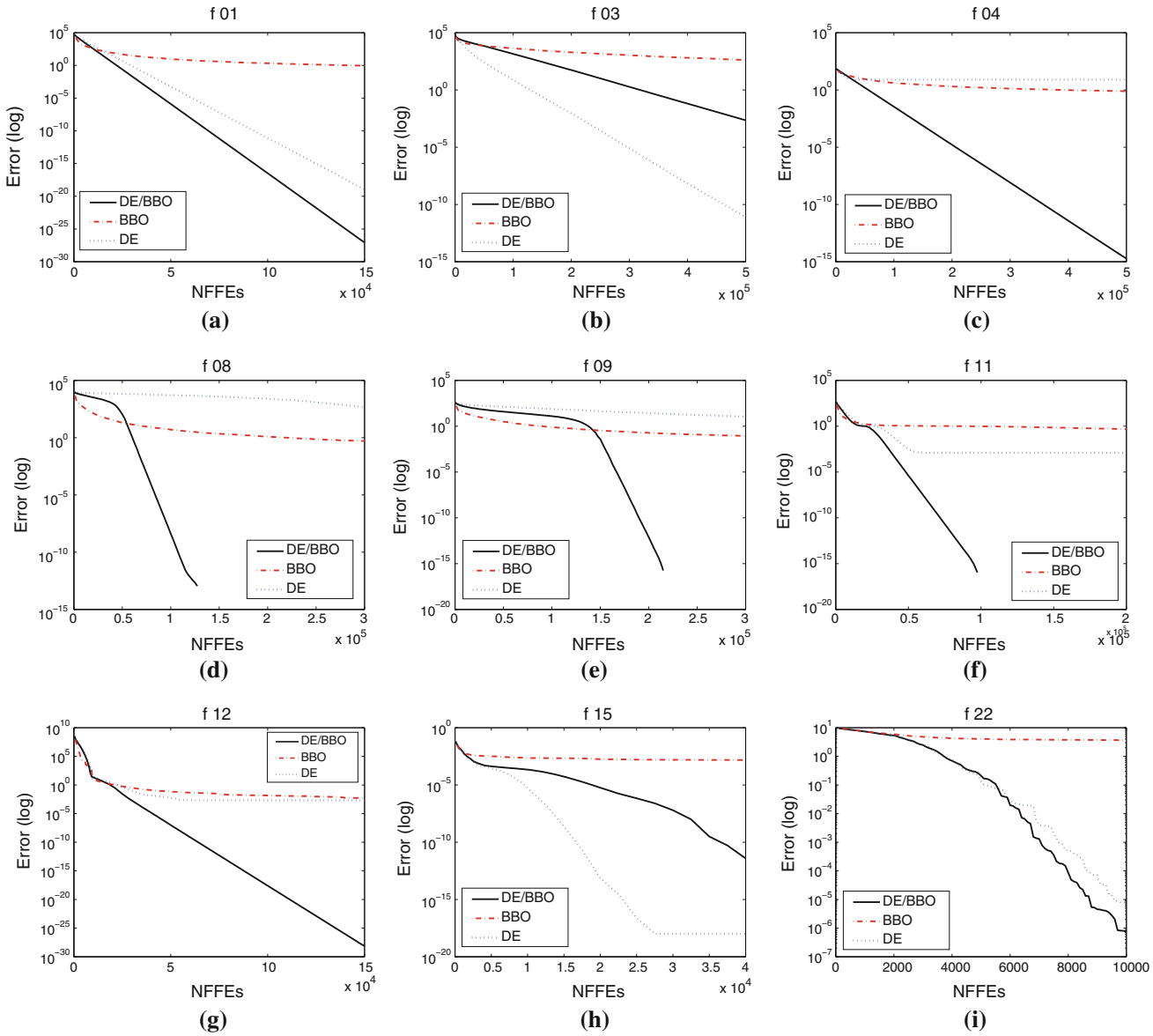

Fig. 1 Mean error curves of DE/BBO, DE, and BBO for selected functions. **a** f01, **b** f03, **c** f04, **d** f08, **e** f09, **f** f11, **g** f12, **h** f15, **i** f22

However, DE may trap into the local minima for five out of six functions. This indicates that our approach has the ability to escape from poor local optima and locate a good near-global optimum. Apparently, from Table 4 it can be seen that DE/BBO requires less NFFEs to reach the VTR than DE on 18 functions. DE is faster than DE/BBO on the rest 5 functions. Additionally, for the majority of the test functions DE/BBO converges faster than DE as shown in Fig. 1.

5.3.2 When compared with BBO

From Tables 3, 4 and Fig. 1, it is obvious that DE/BBO performs significantly better than BBO consistently with respect to all five criteria for all test functions. By carefully looking at Fig. 1, we can see that in the beginning of the evolutionary process BBO converges faster than DE/BBO while DE/BBO is able to improve its solution steadily for a long run. The reason might be that BBO has a good exploitation but lacks the exploration. However, for DE/BBO with the hybrid migration operator, it can balance the exploration and the exploitation effectively.

In general, the performance of DE/BBO is highly competitive with DE, especially for the high-dimensional problems. Moreover, DE/BBO is significantly better than BBO for all problems. Since for the majority of the low-dimensional functions (f14–f23), both DE/BBO and DE have no significant difference, we will not use these

functions in the following experiments. In addition, we also do not compare the algorithms with BBO in the following experiments.

5.4 Influence of population size

The choice of the best population size of DE is always critical for different problems (Gäperle et al. 2002; Teo 2006; Brest and Maučec 2008). Increasing the population size will increase the diversity of possible movements, promoting the exploration of the search space. However, the probability to find the correct search direction decreases considerably (Feoktistov and Janaqi 2004). The influence of population size is investigated in this section. For both DE/BBO and DE, all the parameter settings are the same as mentioned in Sect. 5.1, only except for $NP = 50$, $NP = 150$, and $NP = 200$.

The results for different population size are shown in Table 5. It can be seen that: (i) for $NP = 50$ DE/BBO is significantly better than DE on 10 functions while it is outperformed by DE for function f03. For f05, DE/BBO is sightly better than DE. And for f13, DE/BBO can locate the near-global optimum over all 50 runs, however, DE traps into the local minima on 14 out of 50 runs. (ii) When the population increases to $NP = 100$, DE can obtain higher overall successful runs than $NP = 50$. DE/BBO is

Table 5 Influence of the performance to different population size for functions f01–f13 ($D = 30$)

F	$NP = 50$		$NP = 100$	
	DE/BBO	DE	DE/BBO	DE
f01	**4.93E−34 ± 2.44E−33** (50)	6.73E−33 ± 9.09E−33 (50)†	**8.66E−28 ± 5.21E−28** (50)	1.10E−19 ± 1.34E−19 (50)†
f02	**0.00E+00 ± 0.00E+00** (50)	1.55E−17 ± 4.49E−17 (50)†	**0.00E+00 ± 0.00E+00** (50)	1.66E−15 ± 8.87E−16 (50)†
f03	7.69E−13 ± 8.68E−13 (50)	**3.03E−17 ± 2.10E−16** (50)☆	2.26E−03 ± 1.58E−03 (0)	**8.19E−12 ± 1.65E−11** (50)☆
f04	**1.37E−06 ± 6.00E−06** (40)	1.72E+01 ± 5.64E+00 (0)†	**1.89E−15 ± 8.85E−16** (50)	7.83E+00 ± 3.78E+00 (0)†
f05	**1.10E+01 ± 5.63E+00** (0)	1.27E+01 ± 7.26E+00 (0)	1.90E+01 ± 7.52E+00 (0)	**8.41E−01 ± 1.53E+00** (6)☆
f06	**[9.39E+03 ± 3.09E+02]** (50)	[1.95E+04 ± 6.20E+03] (50)†	**[2.16E+04 ± 5.73E+02]** (50)	[2.89E+04 ± 2.01E+03] (50)†
f07	**1.64E−03 ± 3.67E−04** (50)	4.19E−03 ± 2.05E−03 (50)†	**3.44E−03 ± 8.27E−04** (50)	3.49E−03 ± 9.60E−04 (50)
f08	**2.37E+01 ± 5.35E+01** (41)	5.57E+02 ± 3.38E+02 (0)†	**0.00E+00 ± 0.00E+00** (50)	4.28E+02 ± 4.69E+02 (1)†
f09	**5.37E−01 ± 7.84E−01** (31)	1.23E+01 ± 4.17E+00 (0)†	**0.00E+00 ± 0.00E+00** (50)	1.14E+01 ± 7.57E+00 (0)†
f10	**4.14E−15 ± 0.00E+00** (50)	7.45E−02 ± 2.55E−01 (46)†	**1.07E−14 ± 1.90E−15** (50)	6.73E−11 ± 2.86E−11 (50)†
f11	**3.45E−04 ± 1.73E−03** (48)	4.83E−03 ± 7.90E−03 (32)†	**0.00E+00 ± 0.00E+00** (50)	1.23E−03 ± 3.16E−03 (43)†
f12	**1.57E−32 ± 0.00E+00** (50)	4.98E−02 ± 1.26E−01 (41)†	**7.16E−29 ± 6.30E−29** (50)	2.07E−03 ± 1.47E−02 (49)
f13	**1.36E−32 ± 7.15E−34** (50)	3.61E+00 ± 1.80E+01 (36)	**9.81E−27 ± 7.10E−27** (50)	7.19E−02 ± 5.09E−01 (49)
F	$NP = 150$		$NP = 200$	
	DE/BBO	DE	DE/BBO	DE
f01	**7.60E−23 ± 3.75E−23** (50)	6.36E−12 ± 4.24E−12 (50)†	**1.91E−16 ± 7.51E−17** (50)	7.24E−08 ± 2.98E−08 (0)†
f02	**0.00E+00 ± 0.00E+00** (50)	1.12E−09 ± 3.67E−10 (50)†	**1.88E−14 ± 4.27E−15** (50)	8.87E−07 ± 2.37E−07 (0)†
f03	8.10E−01 ± 4.63E−01 (0)	**1.16E−07 ± 2.24E−07** (2)☆	1.98E+01 ± 9.76E+00 (0)	**3.27E−05 ± 3.17E−05** (0)☆
f04	**2.33E−13 ± 1.02E−13** (50)	4.74E+00 ± 3.21E+00 (0)†	**6.53E−10 ± 2.04E−10** (50)	2.61E+00 ± 1.95E+00 (0)†
f05	1.99E+01 ± 5.26E−01 (0)	**1.84E+00 ± 1.58E+00** (0)☆	2.11E+01 ± 4.06E−01 (0)	**4.85E+00 ± 1.63E+00** (0)☆
f06	**[2.56E+04 ± 6.27E+02]** (50)	[4.27E+04 ± 1.53E+03] (50)†	**[3.36E+04 ± 6.97E+02]** (50)	[5.78E+04 ± 1.74E+03] (50)†
f07	**3.93E−03 ± 7.69E−04** (50)	4.39E−03 ± 1.10E−03 (50)†	**5.03E−03 ± 1.09E−03** (50)	5.55E−03 ± 1.75E−03 (50)
f08	**0.00E+00 ± 0.00E+00** (50)	2.35E+03 ± 1.13E+03 (0)†	**0.00E+00 ± 0.00E+00** (50)	3.10E+03 ± 1.04E+03 (0)†
f09	**0.00E+00 ± 0.00E+00** (50)	3.27E+01 ± 1.43E+01 (0)†	**0.00E+00 ± 0.00E+00** (50)	4.86E+01 ± 1.21E+01 (0)†
f10	**1.91E−12 ± 5.21E−13** (50)	6.40E−07 ± 1.98E−07 (0)†	**3.11E−09 ± 5.76E−10** (50)	7.04E−05 ± 1.68E−05 (0)†
f11	**0.00E+00 ± 0.00E+00** (50)	2.22E−18 ± 1.57E−17 (50)	**0.00E+00 ± 0.00E+00** (50)	4.93E−04 ± 1.99E−03 (47)
f12	**5.13E−24 ± 3.15E−24** (50)	2.07E−03 ± 1.47E−02 (49)	**1.17E−17 ± 5.01E−18** (50)	1.58E−09 ± 1.01E−09 (50)†
f13	**6.84E−22 ± 5.10E−22** (50)	2.51E−03 ± 1.78E−02 (49)	**1.75E−15 ± 8.37E−16** (50)	9.58E−08 ± 7.78E−08 (0)†

Hereafter, (#) indicates the number of successful runs and $[a \pm b]$ denotes the averaged NFFEs required when the global minimum achieved before using Max_NFFEs for all algorithms

†,☆ The value of t with 49 degrees of freedom is significant at $\alpha = 0.05$ by two-tailed test

☆ means that the corresponding algorithm is better than our proposed DE/BBO method

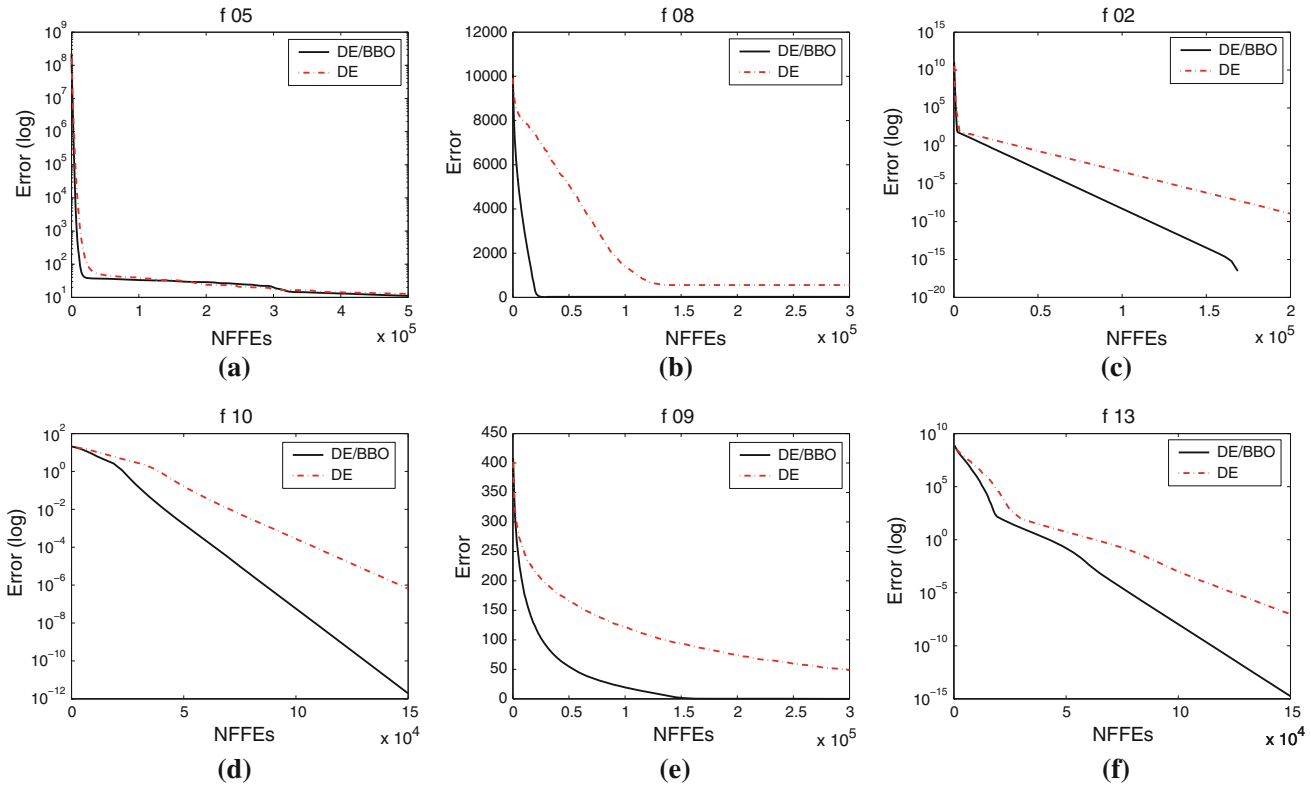

Fig. 2 Mean error curves of DE/BBO and DE to different population size for selected functions at $D = 30$. **a** f05 ($NP = 50$). **b** f08 ($NP = 50$). **c** f02 ($NP = 150$). **d** f10 ($NP = 150$). **e** f09 ($NP = 200$). **f** f13 ($NP = 200$)

significantly better than DE on eight functions based on the *t*-test results. DE is significantly better than DE/BBO for functions f03 and f05. For the rest three functions, DE/BBO is better than DE. (iii) For $NP = 150$ and $NP = 200$, DE/BBO is able to obtain significant better performance than DE on eight and nine functions, respectively. Similarly, for f03 and f05, DE/BBO is outperformed by DE significantly. In addition, from Fig. 2 we can see that DE/BBO can obtain higher convergence speed to different population size for the majority of functions compared with DE.

In general, the overall performance of DE/BBO is better than DE to different population size. DE/BBO exhibits higher overall successful runs, higher convergence velocity, and more robustness than DE.

5.5 Effect of dimensionality

In order to investigate the influence of the problem dimension on the performance of DE/BBO, we carry out a scalability study comparing with the original DE algorithm for the scalable functions in the test suit. For functions f01–f13, $D = 10, 50, 100,$ and 200. The results are recorded in Table 6 after $D \times 10,000$ NFFEs, and some representative convergence graphs are shown in Fig. 3. From Table 6 we can see that the overall *SR* is decreasing for both DE/BBO and DE, since increasing the problem dimension leads to the algorithms sometimes unable to solve the problem before reaching the Max_NFFEs. However, similarly to $D = 30$, on the majority of functions, DE/BBO outperforms DE at every dimension. By carefully looking at the results, we can recognize that for f05 DE is better than DE/BBO at $D = 10$ and $D = 30$, however, DE is outperformed by DE/BBO at higher dimension ($D = 50, 100,$ and 200). So, from the experimental results of this section, we can conclude that the hybrid migration operator has the ability to accelerate DE in general, especially the improvements are more significant at higher dimensionality.

5.6 Influence of different mutation schemes

There are ten mutation schemes proposed in the original DE (Storn and Price 2008; Price et al. 2005). Actually, choosing the best among different mutation schemes available for DE is also not easy for a specific problem (Qin and Suganthan 2005; Qin et al. 2009). In this section, we perform additional experiment to compare the

Table 6 Scalability study for functions f01–f13 at Max_NFFEs $= D \times 10,000$

F	$D = 10$		$D = 50$	
	DE/BBO	DE	DE/BBO	DE
f01	**[1.89E+04 ± 3.63E+02]**(50)	[3.08E+04 ± 8.41E+02] (50)†	**0.00E+00 ± 0.00E+00** (50)	1.87E−32 ± 1.64E−32 (50)†
f02	**[2.63E+04 ± 4.19E+02]** (50)	[4.71E+04 ± 7.61E+02] (50)†	**0.00E+00 ± 0.00E+00** (50)	1.44E−17 ± 3.97E−17 (50)†
f03	6.07E−14 ± 9.60E−14 (50)	**1.35E−21 ± 2.49E−21** (50)☆	5.20E+02 ± 2.48E+02 (0)	**1.12E−01 ± 8.06E−02** (0)☆
f04	**1.58E−16 ± 9.88E−17** (50)	1.39E−13 ± 1.22E−13 (50)†	**3.42E−03 ± 2.28E−02** (3)	1.95E+01 ± 4.14E+00 (0)†
f05	3.40E+00 ± 8.03E−01 (0)	**3.53E−11 ± 1.24E−10** (50)☆	**4.43E+01 ± 1.39E+01** (0)	5.33E+01 ± 2.89E+01 (0)
f06	**[6.62E+03 ± 3.28E+02]** (50)	[1.06E+04 ± 5.50E+02] (50)†	**[2.74E+04 ± 6.22E+02]** (50)	[5.57E+04 ± 1.64E+04] (50)†
f07	**1.11E−03 ± 3.96E−04** (50)	1.52E−03 ± 5.97E−04 (50)†	**4.05E−03 ± 9.20E−04** (50)	9.49E−03 ± 3.07E−03 (34)†
f08	**0.00E+00 ± 0.00E+00** (50)	1.51E−11 ± 1.05E−10 (50)	**2.37E+00 ± 1.67E+01** (49)	1.51E+03 ± 9.97E+02 (0)†
f09	**0.00E+00 ± 0.00E+00** (50)	3.75E+00 ± 2.72E+00 (5)†	**0.00E+00 ± 0.00E+00** (50)	2.33E+01 ± 8.27E+00 (0)†
f10	**5.89E−16 ± 0.00E+00** (50)	8.02E−16 ± 8.52E−16 (50)	**5.92E−15 ± 1.79E−15** (50)	3.52E−02 ± 1.74E−01 (48)
f11	**0.00E+00 ± 0.00E+00** (50)	8.54E−03 ± 1.56E−02 (31)†	**0.00E+00 ± 0.00E+00** (50)	5.08E−03 ± 1.62E−02 (9)†
f12	4.71E−32 ± 0.00E+00 (50)	4.71E−32 ± 0.00E+00 (50)	**9.42E−33 ± 0.00E+00** (50)	4.51E−02 ± 1.40E−01 (41)†
f13	1.35E−32 ± 0.00E+00 (50)	1.35E−32 ± 0.00E+00 (50)	**1.35E−32 ± 0.00E+00** (50)	1.34E−01 ± 5.74E−01 (40)
F	$D = 100$		$D = 200$	
	DE/BBO	DE	DE/BBO	DE
f01	**6.16E−34 ± 1.97E−33** (50)	2.30E−31 ± 1.37E−31 (50)†	**3.07E−32 ± 2.48E−32** (50)	1.41E−24 ± 3.14E−24 (50)†
f02	**0.00E+00 ± 0.00E+00** (50)	7.95E−16 ± 1.05E−15 (50)†	**5.83E−17 ± 8.11E−17** (50)	7.38E−09 ± 2.33E−08 (46)†
f03	3.10E+04 ± 1.12E+04 (0)	**1.31E+02 ± 5.76E+01** (0)☆	2.22E+05 ± 5.90E+04 (0)	**3.64E+03 ± 7.60E+02** (0)☆
f04	**2.71E+00 ± 2.58E+00** (0)	3.05E+01 ± 4.00E+00 (0)†	**1.59E+01 ± 3.43E+00** (0)	4.27E+01 ± 4.31E+00 (0)†
f05	**1.19E+02 ± 3.38E+01** (0)	1.76E+02 ± 4.22E+01 (0)†	**2.95E+02 ± 4.48E+01** (0)	4.39E+02 ± 1.11E+02 (0)†
f06	**[4.93E+04 ± 1.11E+03]** (50)	[3.30E+05 ± 1.34E+05] (50)†	**0.00E+00 ± 0.00E+00** (50)	3.00E+00 ± 7.75E+00 (20)
f07	**6.87E−03 ± 1.15E−03** (49)	4.84E−02 ± 2.19E−02 (0)†	**1.56E−02 ± 2.82E−03** (0)	2.19E−01 ± 7.38E−02 (0)†
f08	**7.11E+00 ± 2.84E+01** (47)	6.79E+03 ± 1.07E+03 (0)†	**2.01E+02 ± 1.68E+02** (23)	2.47E+04 ± 2.44E+03 (0)†
f09	**7.36E−01 ± 8.48E−01** (23)	7.28E+01 ± 1.07E+01 (0)†	**1.76E+01 ± 2.89E+00** (0)	2.12E+02 ± 2.12E+01 (0)†
f10	**7.84E−15 ± 7.03E−16** (50)	1.87E+00 ± 5.72E−01 (1)†	**1.09E−14 ± 1.12E−15** (50)	5.30E+00 ± 8.47E−01 (0)†
f11	**0.00E+00 ± 0.00E+00** (50)	8.43E−03 ± 1.71E−02 (36)†	**1.11E−16 ± 0.00E+00** (50)	1.33E−01 ± 2.50E−01 (0)
f12	**4.71E−33 ± 0.00E+00** (50)	8.07E+03 ± 2.82E+04 (26)†	**2.36E−33 ± 0.00E+00** (50)	5.71E+04 ± 7.31E+04 (8)†
f13	**1.76E−32 ± 2.68E−32** (50)	1.85E+03 ± 7.38E+03 (12)	**8.36E−32 ± 1.44E−31** (50)	1.65E+05 ± 2.91E+05 (0)

†,☆ The value of t with 49 degrees of freedom is significant at $\alpha = 0.05$ by two-tailed test

☆ means that the corresponding algorithm is better than our proposed DE/BBO method

performance of DE/BBO with that of DE to different schemes. Four schemes, namely, DE/best/1/bin, DE/rand/2/bin, DE/rand-to-best/1/bin, and DE/best/2/bin are chosen in these experiments. All remaining parameters are the same as mentioned in Sect. 5.1. Table 7 gives the results of DE/BBO and DE for the four schemes. Based on the t test, the results can be summarized as "$w/t/l$", which means that DE/BBO wins in w functions, ties in t functions, and loses in l functions, compared with DE. From Table 7, for DE/best/1/bin, DE/rand/2/bin, DE/rand-to-best/1/bin, and DE/best/2/bin, they are $12/1/0$, $9/2/2$, $8/3/2$, and $10/2/1$, respectively. The results indicate that DE/BBO is able to obtain greater robustness than DE to different mutation schemes on the majority of functions.

5.7 Influence of self-adaptive parameter control

As mentioned above, the choice of the control parameters F and CR is sensitive for different problems (Gäperle et al. 2002). Researchers have proposed the adaptive parameter control of DE, such as (Liu and Lampinen 2005; Brest et al. 2006), and so on. In order to show that DE/BBO can also improve the self-adaptive DE, in this section, we adopt the self-adaptive parameter control proposed in Brest et al. (2006) to replace $F = \text{rndreal}(0.1, 1.0)$ and $CR = 0.9$ in the previous experiments. All other parameter settings are kept unchanged. The results for the self-adaptive DE (SADE) and self-adaptive DE/BBO (SADE/BBO) are given in Table 8.

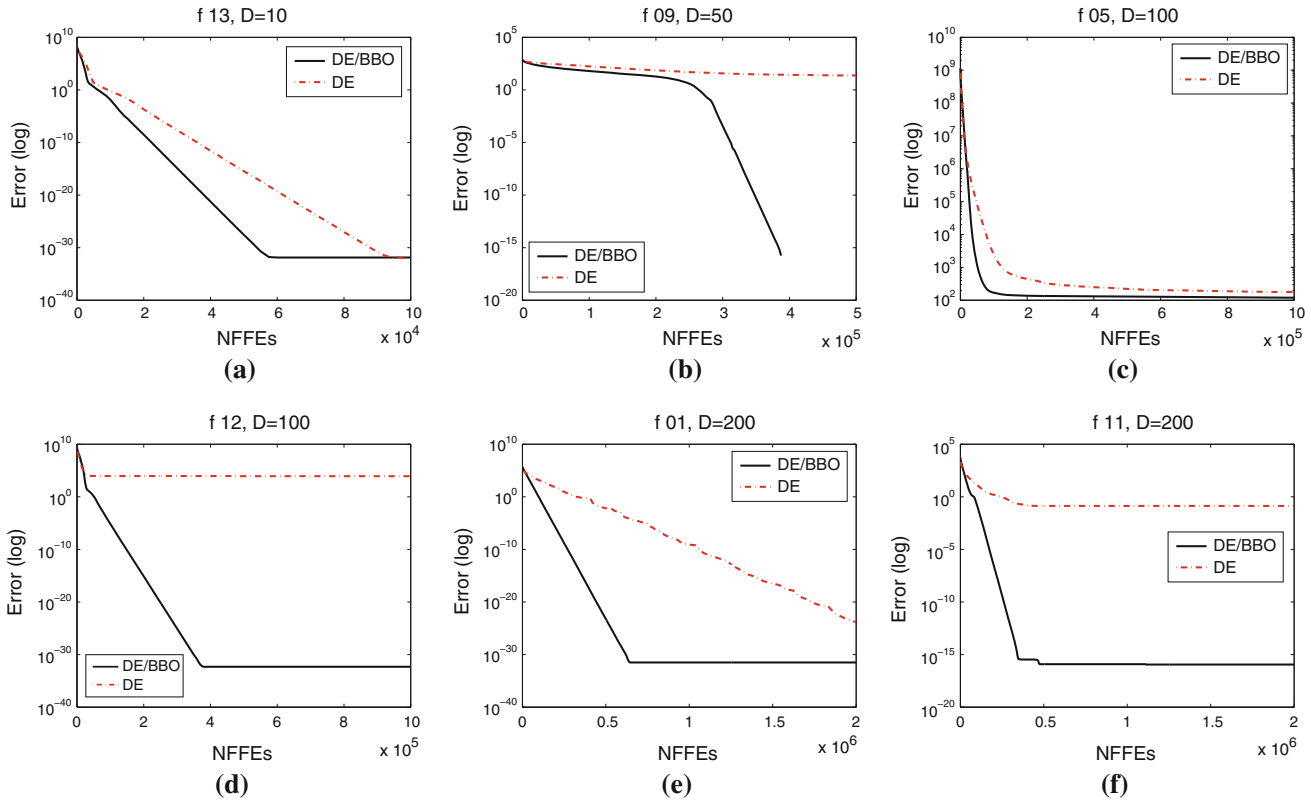

Fig. 3 Mean error curves to compare the scalability of DE/BBO and DE for selected functions. **a** f13 ($D = 10$). **b** f09 ($D = 50$). **c** f05 ($D = 100$). **d** f12 ($D = 100$). **e** f01 ($D = 200$). **f** f11 ($D = 200$)

According to Table 8, we can see that: first, for the Error values, both SADE/BBO and SADE can obtain the global optimum on five functions (f02, f06, f08, f09, and f11) over 50 runs. SADE/BBO is significantly better than SADE on five functions. SADE outperforms SADE/BBO on two functions (f03 and f05). Second, with respect to the NFFEs, it is obvious that SADE/BBO is significantly better than SADE on 11 functions. And the *AR* values are larger than one for these functions, it means that SADE/BBO is faster than SADE. For functions f03 and f05, SADE/BBO fails to solve the two functions over all 50 runs. Overall, integration of the hybrid migration operator can improve the performance of SADE.

5.8 Comparison with other DE hybrids

In this section, we make a comparison with other DE hybrids. Since there are many variants of DE, we only compare our approach with DEahcSPX proposed in Noman and Iba (2008), ODE proposed in Rahnamayan et al. (2008), and DE/EDA proposed in Sun et al. (2005).

5.8.1 Comparison with DEahcSPX and ODE

Firstly, we compare our approach with DEahcSPX and ODE. In DEahcSPX, a crossover-based adaptive local search operation to accelerate the original DE. The authors concluded that DEahcSPX outperforms the original DE in items of convergence rate in all experimental studies. In ODE, the opposition-based learning is used for the population initialization and generation jumping. In this section, we compare our proposed CDE with the original DE, DEahcSPX and ODE. All the parameter settings are the same as mentioned in Sect. 5.1. For DEahcSPX, the number of parents in SPX sets to be $n_p = 3$ (Noman and Iba 2008). For ODE, the jump rate $J_r = 0.3$ (Rahnamayan et al. 2008). The results are given in Table 9. The selected representative convergence graphs are shown in Fig. 4.

It can be seen that, from Table 9, DE/BBO is significantly better than DEahcSPX on eight functions while it is outperformed by DEahcSPX on two functions (f03 and f05). For the rest three functions, there are no significant difference between DE/BBO and DEachSPX. However, for f12 and f13, DE/BBO can obtain the near-global optimum over all 50 runs while DEahcSPX traps into the local minima on one run, respectively. In addition, Fig. 4 shows that DE/BBO converges faster than DEahcSPX on the major functions.

With respect to ODE, the *t* test is 9/2/2 in Table 9. It means that DE/BBO significantly outperforms ODE on

Table 7 Comparison of DE/BBO and DE to different mutation schemes for functions f01–f13 ($D = 30$)

F	DE/best/1/bin		DE/rand/2/bin	
	DE/BBO	DE	DE/BBO	DE
f01	**3.71E-32 ± 3.61E-32** (50)	1.25E+02 ± 1.55E+02 (0)†	**1.27E-17 ± 6.15E−18** (50)	1.53E −10 ± 8.92E−11 (50)†
f02	**5.46E−16 ± 1.81E−15** (50)	4.16E+00 ± 3.02E+00 (0)†	**4.70E−15 ± 1.54E−15** (50)	2.02E−08 ± 9.61E−09 (2)†
f03	**8.15E−30 ± 2.66E−29** (50)	2.62E+02 ± 3.04E+02 (0)†	1.31E+01 ± 7.15E+00 (0)	**7.28E−07 ± 8.90E−07** (0)☆
f04	**2.36E+01 ± 5.72E+00** (0)	2.97E+01 ± 6.06E+00 (0)†	**2.04E−09 ± 7.53E−10** (50)	5.18E+00 ± 3.41E+00 (0)†
f05	**8.72E+00 ± 1.43E+01** (8)	1.14E+04 ± 1.43E+04 (0)†	9.25E+00 ± 1.30E+00 (0)	**9.60E−02 ± 5.63E−01** (0)☆
f06	**1.56E+01 ± 2.98E+01** (0)	1.17E+03 ± 5.42E+02 (0)†	**[3.16E+04 ± 8.95E+02]** (50)	[4.74E+04 ± 1.88E+03] (50)†
f07	**4.07E−03 ± 1.39E−03** (50)	9.53E−03 ± 5.73E−03 (43)†	**5.21E−03 ± 1.26E−03** (50)	5.41E−03 ± 1.46E−03 (49)
f08	**6.58E+02 ± 2.94E+02** (0)	4.84E+03 ± 6.80E+02 (0)†	**0.00E+00 ± 0.00E+00** (50)	6.71E+03 ± 2.95E+02 (0)†
f09	**1.85E+01 ± 7.55E+00** (0)	7.26E+01 ± 1.51E+01 (0)†	**7.60E−05 ± 2.98E−04** (12)	1.16E+02 ± 2.28E+01 (0)†
f10	**2.48E+00 ± 1.16E+00** (2)	9.73E+00 ± 1.66E+00 (0)†	**8.65E−10 ± 2.00E−10** (50)	3.11E−06 ± 1.04E−06 (0)†
f11	**3.22E−02 ± 4.06E−02** (12)	2.13E+00 ± 1.33E+00 (0)†	**0.00E+00 ± 0.00E+00** (50)	6.41E−04 ± 2.22E−03 (46)†
f12	**6.70E−01 ± 1.15E+00** (20)	2.77E+01 ± 2.24E+01 (0)†	**1.31E−18 ± 1.09E−18** (50)	8.03E−12 ± 1.04E−11 (50)†
f13	**7.14E−01 ± 1.24E+00** (1)	2.67E+04 ± 1.28E+05 (0)	**3.28E−16 ± 3.02E−16** (50)	1.29E−04 ± 9.08E−04 (44)

F	DE/rand-to-best/1/bin		DE/best/2/bin	
	DE/BBO	DE	DE/BBO	DE
f01	**0.00E+00 ± 0.00E+00** (50)	1.23E−34 ± 6.10E−34 (50)	**1.65E−32 ± 1.37E−32** (50)	1.65E−31 ± 1.93E−31 (50)†
f02	**[3.42E+04 ± 3.97E+02]** (50)	[4.14E+04 ± 3.72E+03] (50)†	**3.11E−17 ± 6.65E−17** (50)	7.88E−15 ± 3.26E−14 (50)
f03	3.71E−25 ± 8.69E−25 (50)	**5.63E−32 ± 2.72E−32** (50)☆	3.28E−30 ± 5.47E−30 (50)	**1.07E−30 ± 8.69E−31** (50)☆
f04	**2.63E+00 ± 1.35E+00** (0)	2.95E+00 ± 1.31E+00 (0)	**2.31E−06 ± 6.77E−06** (3)	3.64E−02 ± 4.42E−02 (0)†
f05	1.28E+01 ± 3.24E+00 (0)	**1.04E+00 ± 1.77E+00** (37)☆	**1.10E+01 ± 5.63E+00** (0)	1.27E+01 ± 7.26E+00 (0)
f06	**2.00E−02 ± 1.41E−01** (49)	3.42E+00 ± 4.13E+00 (5)†	**2.20E−01 ± 4.65E−01** (40)	1.34E+01 ± 1.95E+01 (2)†
f07	**8.43E−04 ± 2.62E−04** (50)	2.17E−03 ± 7.98E−04 (50)†	**2.20E−03 ± 7.68E−04** (50)	5.33E−03 ± 2.81E−03 (46†
f08	**4.97E+01 ± 7.60E+01** (33)	3.15E+03 ± 5.79E+02 (0)†	**2.61E+01 ± 5.50E+01** (40)	4.60E+03 ± 5.21E+02 (0)†
f09	**3.98E−02 ± 1.97E−01** (48)	2.37E+01 ± 7.13E+00 (0)†	**1.11E+00 ± 1.25E+00** (19)	5.71E+01 ± 1.48E+01 (0)†
f10	**6.13E−15 ± 1.78E−15** (50)	1.69E+00 ± 7.01E−01 (0)†	**1.77E−01 ± 4.19E−01** (42)	3.99E+00 ± 1.14E+00 (0)†
f11	**1.23E−03 ± 3.24E−03** (43)	1.35E−02 ± 1.52E−02 (12)†	**6.74E−03 ± 8.82E−03** (23)	2.60E−02 ± 4.33E−02 (16)†
f12	**2.07E−03 ± 1.47E−02** (49)	7.06E−02 ± 1.71E−01 (34)†	**6.43E−02 ± 1.63E−01** (40)	1.54E+00 ± 2.35E+00 (8)†
f13	**2.20E−04 ± 1.55E−03** (49)	2.71E+00 ± 1.05E+01 (16)	**5.47E−03 ± 1.93E−02** (41)	1.99E+01 ± 2.41E+01 (0)†

†,☆ The value of t with 49 degrees of freedom is significant at $\alpha = 0.05$ by two-tailed test

☆ means that the corresponding algorithm is better than our proposed DE/BBO method

nine out of 13 functions. ODE is significantly better than DE/BBO on two functions (f03 and f07). For f01 and f10, there are no significant difference between DE/BBO and ODE, however, DE/BBO is slightly better than ODE. Moreover, on the majority of functions, DE/BBO exhibits higher convergence rate than ODE.

5.8.2 Comparison with DE/EDA

Secondly, the comparison between DE/BBO and DE/EDA is evaluated in this section. The reason is that DE/BBO is similar to DE/EDA, i.e., both algorithms combine DE with another global optimization algorithm to improve the performance of DE. DE/EDA combines global information extracted by EDA with differential information obtained by DE to create promising solutions (Sun et al. 2005). DE/BBO integrates the migration operator of BBO into DE to balance the exploration and the exploitation. In original DE/EDA algorithm,[3] the DE mutation scheme is described as follows

$$U_i(j) = [X_{r_1}(j) + X_i(j)]/2 + F \times [(X_{r_1}(j) - X_i(j)) + (X_{r_2}(j) - X_{r_3}(j))] \quad (6)$$

where X_i is the target vector of DE. In order to make a fair comparison, all compared algorithms (DE, DE/BBO, and DE/EDA) adopt the mutation scheme shown in Eq. 6 to replace the DE/rand/1/bin scheme. All other parameters are

[3] The source code of DE/EDA is available online at: http://cswww.essex.ac.uk/staff/qzhang/IntrotoResearch/HybridEDA.htm.

Table 8 Influence of self-adaptive parameter control to DE/BBO and DE for functions f01–f13 ($D = 30$)

F	Error		NFFEs		
	SADE/BBO	SADE	SADE/BBO	SADE	AR
f01	**0.00E+00 ± 0.00E+00** (50)	1.75E−27 ± 1.57E−27 (50)†	**3.91E+04 ± 8.15E+02**	6.11E+04 ± 1.12E+03†	1.56
f02	0.00E+00 ± 0.00E+00 (50)	0.00E+00 ± 0.00E+00 (50)	**5.12E+04 ± 8.17E+02**	8.45E+04 ± 1.40E+03†	1.65
f03	2.10E−01 ± 2.85E−01 (0)	**4.03E−13 ± 6.20E−13** (50)☆	NA	**3.57E+05 ± 1.84E+04**	NA
f04	**4.11E−16 ± 1.10E−15** (50)	3.44E−14 ± 1.83E−13 (50)	**2.25E+05 ± 3.57E+04**	3.09E+05 ± 4.54E+03†	1.38
f05	4.05E+01 ± 2.30E+01 (0)	**9.99E−02 ± 1.23E−01** (1)☆	NA	**4.81E+05 ± 0.00E+00**	NA
f06	0.00E+00 ± 0.00E+00 (50)	0.00E+00 ± 0.00E+00 (50)	**1.46E+04 ± 4.91E+02**	2.30E+04 ± 7.05E+02†	1.57
f07	**1.98E−03 ± 4.35E−04** (50)	3.46E−03 ± 9.00E−04 (50)†	**6.33E+04 ± 1.32E+04**	1.11E+05 ± 2.23E+04†	1.75
f08	0.00E+00 ± 0.00E+00 (50)	0.00E+00 ± 0.00E+00 (50)	**4.87E+04 ± 1.26E+03**	9.58E+04 ± 2.27E+03†	1.97
f09	0.00E+00 ± 0.00E+00 (50)	0.00E+00 ± 0.00E+00 (50)	**6.45E+04 ± 3.23E+03**	1.19E+05 ± 4.08E+03†	1.85
f10	**4.14E−15 ± 0.00E+00** (50)	1.54E−14 ± 5.48E−15 (50)†	**5.92E+04 ± 8.21E+02**	9.31E+04 ± 1.63E+03†	1.57
f11	0.00E+00 ± 0.00E+00 (50)	0.00E+00 ± 0.00E+00 (50)	**4.04E+04 ± 7.94E+02**	6.47E+04 ± 3.14E+03†	1.60
f12	**1.57E−32 ± 0.00E+00** (50)	1.15E−28 ± 1.15E−28 (50)†	**3.56E+04 ± 8.09E+02**	5.52E+04 ± 1.28E+03†	1.55
f13	**1.35E−32 ± 0.00E+00** (50)	3.92E−26 ± 5.22E−26 (50)†	**4.22E+04 ± 8.62E+02**	6.71E+04 ± 1.45E+03†	1.59

†,☆ The value of t with 49 degrees of freedom is significant at $\alpha = 0.05$ by two-tailed test

☆ means that the corresponding algorithm is better than our proposed DE/BBO method

Table 9 Comparison the performance of DE/BBO with DE, DEachSPX, and ODE for functions f01–f13 ($D = 30$)

F	DE/BBO	DE	DEahcSPX	ODE
f01	**8.66E−28 ± 5.21E−28** (50)	1.10E−19 ± 1.34E−19 (50)†	2.90E−20 ± 2.28E−20 (50)†	4.33E−25 ± 1.86E−24 (50)
f02	**0.00E+00 ± 0.00E+00** (50)	1.66E−15 ± 8.87E−16 (50)†	4.47E−16 ± 3.66E−16 (50)†	2.81E−13 ± 1.74E−13 (50)†
f03	2.26E−03 ± 1.58E−03 (0)	8.19E−12 ± 1.65E−11 (50)☆	**5.11E−12 ± 9.27E−12** (50)☆	2.50E−11 ± 3.91E−11 (50)☆
f04	**1.89E−15 ± 8.85E−16** (50)	7.83E+00 ± 3.78E+00 (0)†	7.79E+00 ± 3.18E+00 (0)†	9.44E−02 ± 2.33E−01 (14)†
f05	1.90E+01 ± 7.52E+00 (0)	**8.41E−01 ± 1.53E+00** (6)☆	1.24E+00 ± 1.67E+00 (5)☆	2.80E+01 ± 9.24E+00 (0)†
f06	**[2.16E+04 ± 5.73E+02]** (50)	[2.89E+04 ± 2.01E+03] (50)†	[2.81E+04 ± 1.50E+03] (50)†	[2.29E+04 ± 1.81E+03] (50)†
f07	3.44E−03 ± 8.27E−04 (50)	3.49E−03 ± 9.60E−04 (50)	3.52E−03 ± 1.20E−03 (50)	**1.03E−03 ± 3.38E−04** (50)☆
f08	**0.00E+00 ± 0.00E+00** (50)	4.28E+02 ± 4.69E+02 (1)†	4.98E+02 ± 8.42E+02 (5)†	1.63E+03 ± 1.27E+03 (1)†
f09	**0.00E+00 ± 0.00E+00** (50)	1.14E+01 ± 7.57E+00 (0)†	1.30E+01 ± 8.11E+00 (0)†	1.65E+01 ± 1.17E+01 (0)†
f10	**1.07E−14 ± 0.00E+00** (50)	6.73E−11 ± 2.86E−11 (50)†	3.89E−11 ± 1.97E−11 (50)†	5.34E−07 ± 3.77E−06 (49)
f11	**0.00E+00 ± 0.00E+00** (50)	1.23E−03 ± 3.16E−03 (43)†	1.82E−03 ± 5.09E−03 (42)†	2.12E−03 ± 4.66E−03 (39)†
f12	**7.16E−29 ± 6.30E−29** (50)	2.07E−03 ± 1.47E−02 (49)	6.22E−03 ± 2.49E−02 (47)	3.44E−18 ± 1.95E−17 (50)†
f13	**9.81E−27 ± 7.10E−27** (50)	7.19E−02 ± 5.09E−01 (49)	3.22E−02 ± 2.26E−01 (46)	2.05E−22 ± 1.44E−21 (50)†

†,☆ The value of t with 49 degrees of freedom is significant at $\alpha = 0.05$ by two-tailed test ☆ means that the corresponding algorithm is better than our proposed DE/BBO method

the same as mentioned in Sect. 5.1. The results are presented in Table 10. And the selected representative convergence graphs are shown in Fig. 5.

5.8.3 When compared with DE

DE/BBO is significantly better than DE on 11 functions. For the rest two functions (f06 and f13), DE/BBO is also better than DE. Additionally, DE/BBO is able to obtain faster convergence velocity than DE for all functions.

5.8.4 When compared with DE/EDA

The overall *SR* of DE/BBO is better than DE/EDA. On nine functions, DE/BBO is significantly better than DE/EDA. DE/EDA is significantly better than DE/BBO only on one functions (f03). For the rest three functions, there are no significant difference. By carefully looking at the results in Table 10, we can see that DE/BBO is substantial better than DE/EDA for all multimodal functions (f08–f13). DE/BBO can locate the near-global optimum over all 50 runs

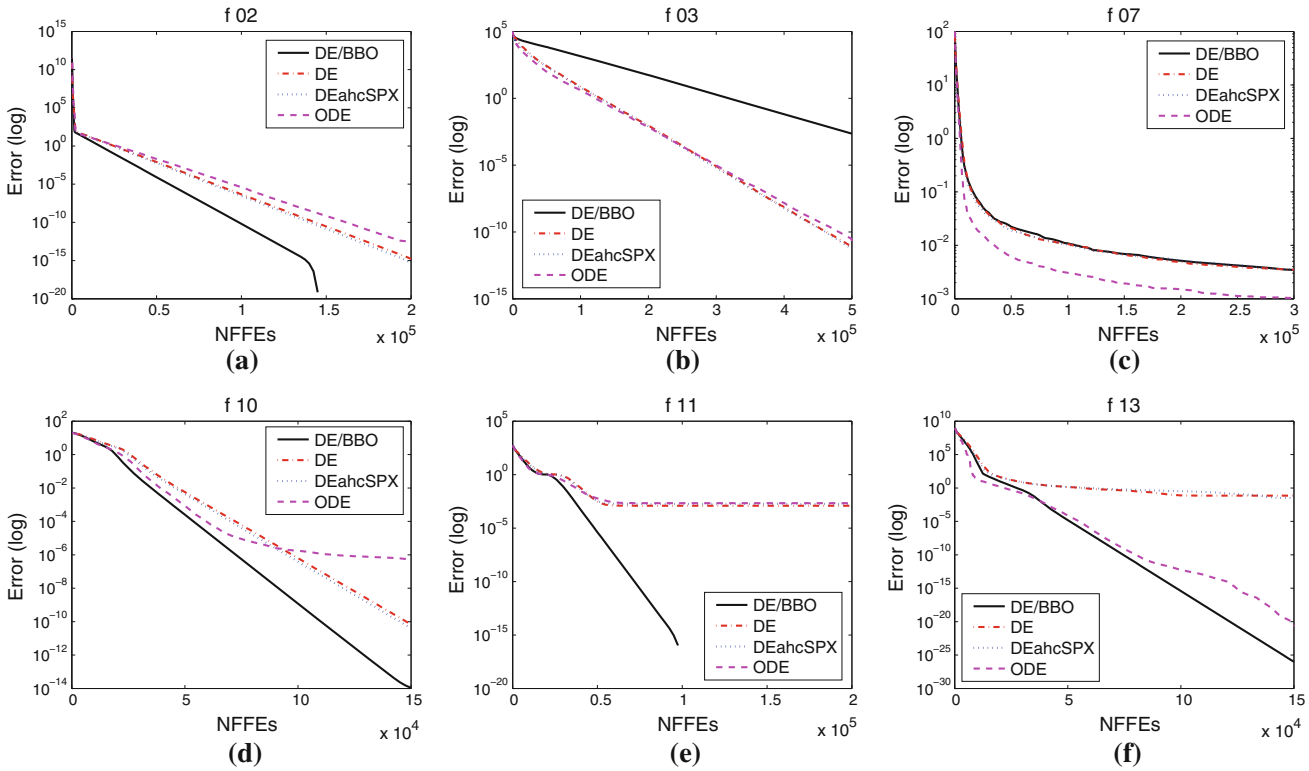

Fig. 4 Mean error curves of DE/BBO, DE, DEahcSPX, and ODE for selected functions. **a** f02, **b** f03, **c** f07, **d** f10, **e** f11, **f** f13

Table 10 Comparison the performance of DE/BBO with DE, and DE/EDA for functions f01–f13 ($D = 30$)

F	DE/BBO	DE	DE/EDA
f01	**0.00E+00 ± 0.00E+00** (50)	2.20E−08 ± 2.91E−08 (22)†	4.68E−25 ± 1.33E−24 (50)†
f02	**0.00E+00 ± 0.00E+00** (50)	8.46E−11 ± 8.62E−11 (50)†	8.33E−16 ± 2.85E−16 (50)†
f03	1.97E−06 ± 2.14E−06 (0)	3.93E−03 ± 5.06E−03 (0)†	**5.27E−16 ± 1.17E−15** (50)☆
f04	**1.46E+00 ± 1.01E+00** (0)	1.14E+01 ± 3.05E+00 (0)†	6.58E+00 ± 1.65E+00 (0)†
f05	2.08E+01 ± 7.69E+00 (0)	3.53E+01 ± 2.64E+01 (0)†	**1.97E+01 ± 1.72E+01** (0)
f06	**0.00E+00 ± 0.00E+00** (50)	2.00E−02 ± 1.41E−01 (49)	**0.00E+00 ± 0.00E+00** (50)
f07	**1.09E−03 ± 3.31E−04** (50)	1.04E−02 ± 3.70E−03 (23)†	3.10E−03 ± 1.31E−03 (50)†
f08	**0.00E+00 ± 0.00E+00** (50)	6.53E+03 ± 4.96E+02 (0)†	7.81E+03 ± 2.77E+02 (0)†
f09	**4.57E−12 ± 2.91E−11** (50)	2.48E+01 ± 2.32E+01 (0)†	7.72E+00 ± 2.52E+00 (0)†
f10	**4.07E−15 ± 5.02E−16** (50)	1.09E+00 ± 7.27E−01 (0)†	1.26E+00 ± 6.31E−01 (7)†
f11	**0.00E+00 ± 0.00E+00** (50)	1.28E−02 ± 1.70E−02 (24)†	1.67E−02 ± 1.91E−02 (16)†
f12	**1.57E−32 ± 0.00E+00** (50)	3.73E−02 ± 9.54E−02 (31)†	3.73E−02 ± 9.07E−02 (38)†
f13	**1.35E−32 ± 0.00E+00** (50)	1.18E+02 ± 7.77E+02 (0)	6.53E−01 ± 4.17E+00 (39)

†,☆ The value of t with 49 degrees of freedom is significant at $\alpha = 0.05$ by two-tailed test

☆ means that the corresponding algorithm is better than our proposed DE/BBO method

for all these functions. However, DE/EDA traps into the local minima many times. Especially, for f08 and f09, DE/EDA fails to solve the two functions. In addition, Fig. 5 shows that DE/BBO converges faster than DE/EDA on the major functions.

5.9 Non-parametric statistical tests

In the previous sections, we presented the experimental results with paired t test, which is a parametric statistical test. However, recent studies indicated that the parametric

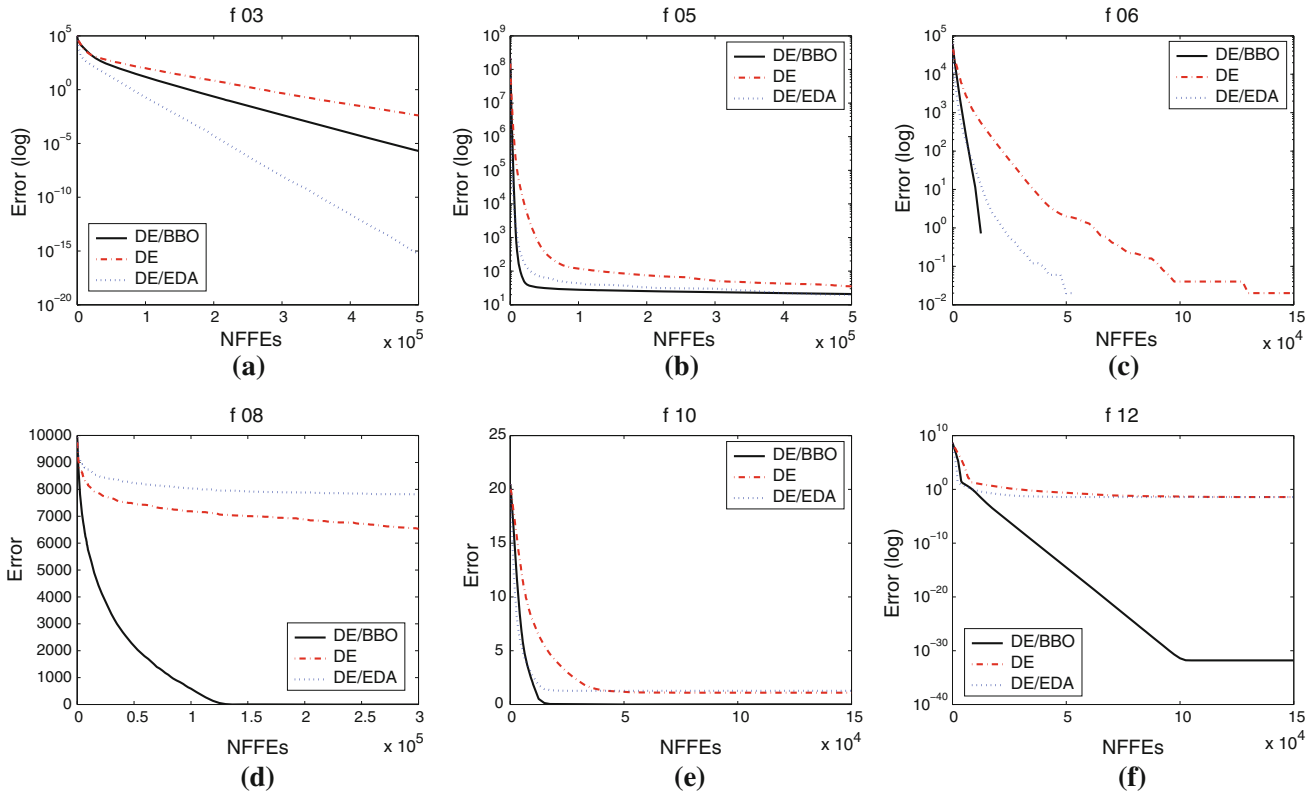

Fig. 5 Mean error curves of DE/BBO, DE, and DE/EDA for selected functions. **a** f03, **b** f05, **c** f06, **d** f08, **e** f10, **f** f12

Table 11 Results of the single-problem non-parametric statical Wilcoxon's test at $\alpha = 0.05$ for functions f01–f13 ($D = 30$)

	BBO		DE		DEahcSPX		ODE		DE/EDA	
	p value	Significant	p value	Significant	p value	Significant	p value	Significant	p value	Significant
f01	0.00	+	0.00	+	0.00	+	0.00	+	0.00	+
f02	0.00	+	0.00	+	0.00	+	0.00	+	0.00	+
f03	0.00	+	0.00	−	0.00	−	0.00	−	0.00	−
f04	0.00	+	0.00	+	0.00	+	0.00	+	0.00	+
f05	0.00	+	0.00	−	0.00	−	0.00	+	0.22	≈
f06	0.00	+	1.00	≈	1.00	≈	1.00	≈	1.00	≈
f07	0.00	+	0.99	≈	1.00	≈	0.00	−	0.00	+
f08	0.00	+	0.00	+	0.00	+	0.00	+	0.00	+
f09	0.00	+	0.00	+	0.00	+	0.00	+	0.00	+
f10	0.00	+	0.00	+	0.00	+	0.00	+	0.00	+
f11	0.00	+	0.02	+	0.01	+	0.00	+	0.00	+
f12	0.00	+	0.00	+	0.00	+	0.00	+	0.00	+
f13	0.00	+	0.00	+	0.00	+	0.00	+	0.00	+

Hereafter, " + ", " − ", and "≈ " indicate that DE/BBO is significantly better, significantly worse, and indifferent, respectively, compared with other algorithms

statistical analysis is not appreciate especially when we tackle the multiple-problem results (Demsar 2006; García and Herrera 2008; García et al. 2009a, b). In order to further prove statistical significance of the results, in this section, we adopt the non-parametric statistical test to compare the DE/BBO method with other algorithms. There are many types of non-parametric statistical methods (García et al. 2009a, b), such as Bonferroni–Dunn's procedure, Holm procedure, Hochberg procedure, Wilcoxon's test, and so on. In this study, the Wilcoxon's test is

Table 12 Results of the multiple-problem non-parametric statical Wilcoxon's test at $\alpha = 0.05$ for functions f01–f13 ($D = 30$)

	R^+	R^-	p value	Significant
BBO	91	0	0.00	+
DE	60	18	0.11	≈
DEahcSPX	61	17	0.09	≈
ODE	63	15	0.06	≈
DE/EDA	67	11	0.03	+

employed since this test is included in well-known software packages (e.g., SPSS, SAR, OriginPro, Matlab, etc.).

To apply the Wilcoxon's test the results are obtained from the previous experiments at $D = 30$ for functions f01–f13. The results of DE/BBO is compared with those of BBO, DE, DEahcSPX, ODE, and DE/EDA. First, we present the results of the single-problem Wilcoxon's test at $\alpha = 0.05$ in Table 11. From Table 11, we can see that DE/BBO is significantly better than the other algorithms in 51 out of 65 cases. In seven cases, DE/BBO is outperformed by the other algorithms. For the remaining seven cases, there is no significance. In summary, the DE/BBO approach outperforms the other algorithms in 78.5% of the cases and is outperforms in only 10.8% of the cases.

Secondly, we present the results of the multiple-problem Wilcoxon's test at $\alpha = 0.05$ in Table 12. It is clear that DE/BBO obtains higher R^+ values than R^- values in all cases. According to the p value, we can see that DE/BBO is significantly better than BBO and DE/EDA. However, with a $\alpha = 0.10$, DE/BBO is able to provide different results compared with BBO, DEahcSPX, ODE, and DE/EDA.

5.10 Experimental results on CEC'05 test functions

In order to further evaluate the performance of DE/BBO, in this section, we report the results of SADE/BBO and SADE for CEC'05 test functions (Suganthan et al. 2005) at $D = 10$. From the experimental results described in Sect. 5.7 showed that the parameter self-adaptation proposed in (Brest et al. 2006) is able to enhance both of the performance of DE/BBO and DE; therefore, this parameter self-adaptation is used. In addition, since most of CEC'05 test functions are rotated, in SADE/BBO for each individual $\lambda_i = 1.0$. All other parameter settings are kept unchanged as Sect. 5.1. The results are summarized in Tables 13 and 14. All results are averaged over 50 independent runs when Max_NFFEs = 100,000.

From Table 13 we can see that SADE/BBO is significantly better than SADE on 13 out of 25 functions. On two functions (f15 and f23), SADE/BBO is outperformed by SADE. On the rest 10 functions, there is no significant difference between SADE/BBO and SADE in terms of the best error values. The results shown in Table 14 indicate that SADE/BBO is able to obtain faster convergence rate on the successful functions. The reason is that the enhanced exploitation by the hybrid migration operator can accelerate the convergence rate. However, due to the enhanced exploitation, SADE/BBO gets stuck in the local minima of some very complex functions, e.g., f15 and f23. The population restart technique (Auger and Hansen 2004) might be used to remedy the premature convergence of SADE/BBO. We will verify it in our future work. In general, our proposed SADE/BBO obtains better results than SADE on

Table 13 Best error values achieved when Max_NFFEs = 100,000 for CEC'05 test functions f01–f25 ($D = 10$)

F	SADE/BBO	SADE	F	SADE/BBO	SADE
f01	0.00E+00 ± 0.00E+00 (50)	0.00E+00 ± 0.00E+00 (50)	f14	**2.70E+00 ± 4.66E−01 (0)**	3.34E+00 ± 1.75E−01 (0)†
f02	**7.40E−29 ± 1.53E−28** (50)	1.10E−15 ± 2.25E−15 (50)†	f15	1.25E+02 ± 1.65E+02 (22)	**7.10E+00 ± 1.85E+01 (43)** ☆
f03	**9.34E−09 ± 6.10E−08 (47)**	9.39E−03 ± 6.30E−02 (9)†	f16	**9.70E+01 ± 1.11E+01 (0)**	1.08E+02 ± 6.30E+00 (0)†
f04	**4.34E−29 ± 7.00E−29 (50)**	2.39E−14 ± 4.96E−14 (50)†	f17	**1.00E+02 ± 1.42E+01 (0)**	1.27E+02 ± 1.02E+01 (0)†
f05	**1.13E−12 ± 4.01E−12 (50)**	7.92E−07 ± 9.28E−07 (0)†	f18	7.36E+02 ± 2.11E+02 (0)	**7.30E+02 ± 1.75E+02 (0)**
f06	4.74E−01 ± 1.12E+00 (39)	**8.07E−02 ± 2.24E−01 (4)**	f19	7.43E+02 ± 1.84E+02 (0)	**7.10E+02 ± 1.94E+02 (0)**
f07	**5.30E−02 ± 3.97E−02 (4)**	5.39E−02 ± 3.32E−02 (2)	f20	**7.05E+02 ± 2.27E+02 (0)**	7.10E+02 ± 1.94E+02 (0)
f08	2.04E+01 ± 7.99E−02 (0)	**2.03E+01 ± 6.33E−02 (0)**	f21	**5.12E+02 ± 1.26E+02 (0)**	7.24E+02 ± 1.88E+02 (0)†
f09	0.00E+00 ± 0.00E+00 (50)	0.00E+00 ± 0.00E+00 (50)	f22	7.59E+02 ± 6.75E+01 (0)	**7.54E+02 ± 6.56E+01 (0)**
f10	**5.52E+00 ± 2.93E+00 (0)**	1.04E+01 ± 2.28E+00 (0)†	f23	7.04E+02 ± 1.91E+02 (0)	**6.43E+02 ± 1.46E+02 (0)** ☆
f11	**1.11E+00 ± 1.85E+00 (1)**	5.90E+00 ± 6.87E−01 (0)†	f24	2.00E+02 ± 0.00E+00 (0)	2.00E+02 ± 0.00E+00 (0)
f12	**1.88E+01 ± 1.00E+02 (32)**	1.95E+01 ± 1.00E+02 (14)†	f25	**5.16E+02 ± 4.99E+00 (0)**	5.17E+02 ± 5.01E+00 (0)†
f13	**4.23E−01 ± 6.08E−02 (0)**	6.04E−01 ± 2.85E−01 (0)†			

The results were averaged over 50 independent runs

† indicates SADE/BBO is significantly better than its competitor by the Wilcoxon signed-rank test at $\alpha = 0.05$

☆ means that the corresponding algorithm is better than our proposed SADE/BBO method

Table 14 Successful NFFEs obtained by SADE/BBO and SADE for CEC'05 test functions f01–f25 ($D = 10$)

F	SADE/BBO	SADE	AR
f01	**1.31E+04 ± 3.37E+02**	2.52E+04 ± 6.57E+02	1.92
f02	**2.90E+04 ± 2.08E+03**	6.42E+04 ± 3.10E+03	2.21
f03	**7.92E+04 ± 1.07E+04**	9.46E+04 ± 1.98E+03	1.19
f04	**3.59E+04 ± 2.25E+03**	6.92E+04 ± 3.43E+03	1.93
f05	**6.80E+04 ± 6.65E+03**	NA ± NA	NA
f06	**6.86E+04 ± 1.19E+04**	9.53E+04 ± 5.16E+03	1.39
f07	**4.03E+04 ± 4.48E+03**	9.27E+04 ± 7.92E+03	2.30
f09	**1.90E+04 ± 7.15E+02**	4.10E+04 ± 1.67E+03	2.16
f12	**4.96E+04 ± 6.47E+03**	9.01E+04 ± 6.05E+03	1.82
f15	**2.04E+04 ± 1.35E+03**	5.79E+04 ± 3.82E+03	2.84

the majority of CEC'05 test functions at $D = 10$ in terms of the quality of the final results and the convergence speed.

5.11 Discussions

The DE algorithm is a fast, robust, and simple global optimization algorithm. However, it may lack the exploitation. BBO is novel optimization algorithm for global optimization. BBO has a good exploitation with the migration operator. Therefore, in this work, we hybridize DE with BBO and propose a hybrid migration operator to generate the promising candidate solution. And then, the DE/BBO algorithm is proposed based on the hybrid migration operator. From the experimental results we can summarize that

- Our proposed DE/BBO approach is effective and efficient. It can obtain the global, or near-global, optimum for the test functions.
- The overall performance of DE/BBO is superior to or highly competitive with BBO and other compared state-of-the-art DE algorithms.
- DE/BBO and DE were compared for different population sizes. On the majority of functions, DE/BBO is substantial better than DE.
- The scalability studies show that DE/BBO is able to accelerate DE in general, especially the improvements are more significant at higher dimensionality.
- Comparison of DE/BBO and DE to different mutation schemes, the overall performance of DE/BBO is more robust than that of DE.
- The self-adaptive parameter control can enhance the performance of DE/BBO and DE. Our proposed hybrid migration operator shows the potential to accelerate the self-adaptive variants of DE.
- According to the non-parametric single/multiple-problem Wilcoxon's test, we can conclude that our approach is better, or at least highly competitive, than the compared algorithm for the test functions.
- Due to the hybrid migration operator proposed in this paper, DE/BBO is able to balance the exploration and the exploitation. In addition, the hybrid migration operator can make the good solutions share more information with the poor ones, meanwhile, it can prevent the good solutions from being destroyed during the evolution. This might be the reason that DE/BBO is significantly better than the original DE algorithm on all tested multimodal functions.
- For function f03, DE/BBO is worse than DE with DE/rand/1/bin scheme. However, from Tables 7 and 10, we can see that DE/BBO is better than DE for DE/best/1/bin and scheme described in Eq. 6. So, we can expect that the strategy adaptation as proposed in Qin et al. (2009) may be used to make DE/BBO more robust.

6 Conclusions and future work

In order to balance the exploration and the exploitation of DE, in this paper, we propose a hybrid DE approach, called DE/BBO, which combines the exploration of DE with the exploitation of BBO. In DE/BBO, a new hybrid migration operator is proposed to generate the promising solutions. Since the hybrid migration operator has a good trade-off between the exploration and the exploitation, it makes our proposed DE/BBO approach be very effective and efficient. To verify the performance of DE/BBO, 23 benchmark functions chosen from literature are employed. Experimental results demonstrate the good performance of our approach. Compared with BBO, DE, DEahcSPX, ODE, and DE/EDA, the results show that DE/BBO is superior to or at least highly competitive with them. Moreover, the influence of the population size, dimensionality, different mutation schemes, and the self-adaptive control parameters of DE/BBO and DE are also investigated. And the results confirm that DE/BBO exhibits a higher convergence rate and greater robustness compared with DE. In addition, compared the results between our approach and SADE on CEC'05 functions, SADE/BBO obtains better results than SADE on the majority of CEC'05 test functions at $D = 10$.

In this work, 23 benchmark functions are used to evaluate the performance of our approach, we will test our approach on more problems, such as the high-dimensional ($D \geq 30$) CEC'05 test suit (Suganthan et al. 2005) and the real-world problems. Moreover, we will compare DE/BBO with other EAs, like the work in García et al. (2009a, b). In addition, we only consider the unconstrained function optimization in this work. Our future work consists on

adding the diversity rules into DE/BBO for constrained optimization problems.

Acknowledgments The authors would like to thank Prof. Brest for providing the SADE code. They are also grateful to the area editor and the anonymous reviewers for their valuable comments and suggestions on this paper. This work was supported in part by the Fund for Outstanding Doctoral Dissertation of China University of Geosciences, China Scholarship Council under Grant No. 2008641008, and the National High Technology Research and Development Program of China under Grant No. 2009AA12Z117.

References

Alatas B, Akin E, Karci A (2008) MODENAR: multi-objective differential evolution algorithm for mining numeric association rules. Appl Soft Comput 8(1):646–656

Auger A, Hansen N (2004) A restart CMA evolution strategy with increasing population size. In: Proceedings of the 2005 IEEE congress on evolutionary computation, pp 1769–1776

Bäck T (1996) Evolutionary algorithms in theory and Practice: evolution strategies, evolutionary programming, genetic algorithms. Oxford University Press, Oxford, UK

Brest J, Maučec MS (2008) Population size reduction for the differential evolution algorithm. Appl Intell 29(3):228–247

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

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

Chakraborty U (2008) Advances in differential evolution. Springer, Berlin

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

Das S, Konar A, Chakraborty UK (2005) Two improved differential evolution schemes for faster global search. In: Beyer HG, O'Reilly UM (eds) Genetic and evolutionary computation conference, GECCO 2005, Proceedings, ACM, Washington DC, USA, June 25–29, 2005, pp 991–998

Demsar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30

Fan HY, Lampinen J (2003) A trigonometric mutation operation to differential evolution. J Global Opt 27(1):105–129

Feoktistov V (2006) Differential evolution: in search of solutions. Springer, New York

Feoktistov V, Janaqi S (2004) Generalization of the strategies in differential evolution. IEEE Comput Soc, Los Alamitos, CA, USA, p 165a

Gäperle R, Müler S, Koumoutsakos P (2002) A parameter study for differential evolution. In: Proceedings of WSEAS International conference on advances in intelligent systems, fuzzy systems, evolutionary computation, pp 293–298

García S, Herrera F (2008) An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. J Mach Learn Res 9:2677–2694

García S, Fernandez A, Luengo J, Herrera F (2009a) A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability. Soft Comput 13(10):959–977. doi:10.1007/s00500-008-0392-y

García S, Molina D, Lozano M, Herrera F (2009b) A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 special session on real parameter optimization. J Heuristics 15:617–644. doi:10.1007/s10732-008-9080-4

Gong W, Cai Z, Ling CX (2006) ODE: afast and robust differential evolution based on orthogonal design. In: AI 2006: advances in artificial intelligence, 19th Australian joint conference on artificial intelligence, Hobart, Australia, December 4–8, 2006, Proceedings, vol 4304. Springer, Berlin, German, pp 709–718

Goulden CH (1956) Methods of statistical analysis, 2nd ed. Wiley, New York

Grosan C, Abraham A, Ishibuchi H (2009) Hybrid evolutionary algorithms. Springer, Berlin, Germany

Hansen N, Kern S (2004) Evaluating the CMA evolution strategy on multimodal test functions. In: Proceedings of the parallel problem solving for nature—PPSN VIII, vol LNCS 3242, pp 282–291

Herrera FML (2000) Two-loop real-coded genetic algorithms with adaptive control of mutation step sizes. Appl Intell 13(3):187–204

Herrera F, Lozano M, Verdegay JL (1998) Tackling real-coded genetic algorithms: operators and tools for behavioural analysis. Artif Intell Rev 12:265–319

Jiao L, Li Y, Gong M, Zhang X (2008) Quantum-inspired immune clonal algorithm for global optimization. IEEE Trans Syst Man Cybern B 38(5):1234–1253

Kaelo P, Ali MM (2007) Differential evolution algorithms using hybrid mutation. Comput Optim Appl 37(2):231–246

Langdon WB, Poli R (2007) Evolving problems to learn about particle swarm optimizers and other search algorithms. IEEE Trans Evol Comput 11(5):561–578

Liang JJ, Qin AK, Suganthan PN, Baskar S (2006) Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. IEEE Trans Evol Comput 10(3):281–295

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

Lozano M, García-Martínez C (2010) Hybrid metaheuristics with evolutionary algorithms specializing in intensification and diversification: overview and progress report. Comput Oper Res 37(3):481–497

Lozano M, Herrera F, Krasnogor N, Molina D (2004) Real-coded memetic algorithms with crossover hill-climbing. Evol Comput 12(3):273–302

Nobakhti A, Wang H (2008) A simple self-adaptive differential evolution algorithm with application on the alstom gasifier. Appl Soft Comput 8(1):350–370

Noman N, Iba H (2005) Enhancing differential evolution performance with local search for high dimensional function optimization. In: Beyer HG, O'Reilly UM (eds) Genetic and evolutionary computation conference, GECCO 2005, Proceedings. ACM, Washington DC, USA, pp 967–974

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

Onwubolu GC, Davendra D (2009) Differential evolution: a handbook for global permutation-based combinatorial optimization. Springer, Berlin

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

Qin AK, Suganthan PN (2005) Self-adaptive differential evolution algorithm for numerical optimization. In: IEEE congress on evolutionary computation (CEC2005), IEEE, pp 1785–1791

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

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

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

Simon D (2008a) Biogeography-based optimization. IEEE Trans Evol Comput 12(6):702–713

Simon D (2008b) The Matlab code of biogeography-based optimization. http://academic.csuohio.edu/simond/bbo/

Storn R, Price K (1997) Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J Global Opt 11(4):341–359

Storn R, Price K (2008) Home page of differential evolution. http://www.ICSI.Berkeley.edu/~storn/code.html

Suganthan PN, Hansen N, Liang JJ, Deb K, Chen YP, Auger A, Tiwari S (2005) Problem definitions and evaluation criteria for the CEC2005 special session on real-parameter optimization. URL http://www.ntu.edu.sg/home/EPNSugan

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

Teng NS, Teo J, Hijazi MHA (2009) Self-adaptive population sizing for a tune-free differential evolution. Soft Comput 13(7):709–724

Teo J (2006) Exploring dynamic self-adaptive populations in differential evolution. Soft Comput 10(8):673–686

Wang YJ, Zhang JS, Zhang GY (2007) A dynamic clustering based differential evolution algorithm for global optimization. Euro J Oper Res 183(1):56–73

Yao X, Liu Y (1997) Fast evolution strategies. Control Cybern 26(3):467–496

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

Zhong W, Liu J, Xue M, Jiao L (2004) A multiagent genetic algorithm for global numerical optimization. IEEE Trans Syst Man Cybern B 34(2):1128–1141