

# A Real-Coded Biogeography-Based Optimization with Neighborhood Search Operator<sup>☆</sup>

Wenyin Gong<sup>a,\*</sup>, Zhihua Cai<sup>a</sup>, Charles X. Ling<sup>b</sup>

<sup>a</sup>*School of Computer Science,  
China University of Geosciences, Wuhan 430074, P.R. China*

<sup>b</sup>*Department of Computer Science,  
The University of Western Ontario, London, Ontario, N6A 5B7, Canada*

---

## Abstract

Biogeography-Based Optimization (BBO) is a new biogeography inspired algorithm for global optimization. There are some open research questions that should be addressed for BBO. In this paper, we extend the original BBO and propose a real-coded BBO approach, referred to as RCBBO, for the global optimization problems in the continuous domain. In addition, in order to enhance the exploration ability of RCBBO, the neighborhood search operator is integrated into it. Experiments have been conducted on 23 benchmark problems of a wide range of dimensions and diverse complexities. The results indicate the good performance of the proposed RCBBO method. Especially, experimental results also show that the neighborhood search operator can improve the exploration ability of RCBBO effectively.

*Key words:*

Biogeography-based optimization, neighborhood search, global optimization, exploration ability, real code

---

## 1. Introduction

The global optimization problems frequently arise in almost every field of engineering design, applied sciences, molecular biology and other scientific applications. Without loss of generality, the unconstrained continuous 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  $S$  [1]. More specifically, it is required to find an  $X^* \in S$  such that

$$\forall X \in S : f(X^*) \leq f(X) \quad (1)$$

where  $f$  does not need to be continuous but it must be bounded.

The major challenge of the continuous global optimization is that the problems to be optimized may have many local optima. This issue is particularly challenging when the dimension is high. During the last few decades, Evolutionary Algorithms (EAs) [2] have been proposed for the continuous global optimization problems. There are many different EAs for global optimization, such as Genetic Algorithms (GAs) [3], Evolution Strategy (ES) [4], Evolutionary Programming (EP) [1], Particle Swarm Optimization (PSO) [5], Differential Evolution (DE) [6], and so on.

Biogeography-Based Optimization (BBO), proposed by Simon [7], is a new global optimization algorithm based on the biogeography theory, which is the study of the geographical distribution of biological organisms. Similar to

---

<sup>☆</sup>This work was supported by the Fund for Outstanding Doctoral Dissertation of CUG, China Scholarship Council under Grant No. 2008641008, the Humanities Base Project of Hubei Province under Grant No. 2004B0011, and the Natural Science Foundation of Hubei Province under Grant No. 2003ABA043.

\*Corresponding author.

*Email addresses:* cug11100304@yahoo.com.cn (Wenyin Gong), zhcai@cug.edu.cn (Zhihua Cai), cling@csd.uwo.ca (Charles X. Ling)

*Preprint submitted to ELsevier*

*April 2, 2009*

GAs, BBO is a population-based, stochastic global optimizer. In the original BBO algorithm, each solution (individual) is a vector of integers. An ensemble of individuals composes a population. Before optimizing, BBO initializes the population randomly. Next, each individual of population is evaluated. And then, BBO uses the migration and mutation operators to update the population. BBO adopts the migration operator to share information between solutions; this feature is similar to other biology-based algorithms, such as GAs and PSO. This 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 [7]. 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. Since BBO is a new global optimization algorithm, there are several open research questions that should be addressed, such as modifying the BBO method so that it could be used to directly optimize functions of continuous variables [7]. In addition, with the probabilistic migration, BBO has a good exploitation ability. However, it lacks the exploration ability.

The aims of this paper are twofold. First, we propose an extension of the original BBO algorithm, namely RCBBO. In RCBBO, each individual is directly encoded by floating point to solve the continuous global optimization problems. Second, the neighborhood search operator is integrated into RCBBO to enhance its exploration ability. Experiments have been conducted on 23 benchmark problems of a wide range of dimensions and diverse complexities. The results indicate the good performance of the proposed RCBBO method. Additionally, experimental results also show that the neighborhood search operator can improve the exploration ability of RCBBO effectively.

The remainder of this paper is organized as follows. In Section 2, the original BBO algorithm is briefly introduced. The neighborhood search operators used in this work are described in Section 3. Section 4 presents our proposed RCBBO approach in detail. This is followed by the performance verification of the proposed approach over 23 benchmark functions in Section 5. The last section, Section 6, is devoted to the conclusions and future work.

## 2. Biogeography-based optimization: BBO

BBO [7] is a new population-based, biogeography inspired global optimization algorithm. 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  $\lambda$  and emigration rate  $\mu$ . A good solution has higher  $\mu$  and lower  $\lambda$ , 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

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

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

where  $I$  is the maximum possible immigration rate;  $E$  is the maximum possible emigration rate;  $k$  is the number of species of the  $k$ -th individual; and  $n$  is the maximum number of species. Note that Eqns. 2 and 3 are just one method for calculating  $\lambda$  and  $\mu$ . There are other different options to assign them based on different specie models [7].

Suppose that we have a global optimization problem and a population of candidate individuals. The individual is represented by a  $D$ -dimensional integer 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 and the mutation operator can be described in Algorithm 1 and Algorithm 2, respectively. Where  $\text{rndreal}(0, 1)$  is a uniformly distributed random real number in  $(0, 1)$  and  $X_i(j)$  is the  $j$ -th SIV of the solution  $X_i$ .  $m_i$  is the mutation rate that is calculated as

$$m_i = m_{max} \left( 1 - \frac{P_i}{P_{max}} \right) \quad (4)$$

where  $m_{max}$  is a user-defined parameter, and  $P_{max} = \text{argmax} P_i, i = 1, \dots, NP$ . 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. Additionally, the

Table 1: The population before the migration operation.

	$X(1)$	$X(2)$	$X(3)$	$X(4)$	$X(5)$	fitness (HSI)	$\lambda$	$\mu$
$X_1$	0	1	0	1	0	2	0.00	1.00
$X_2$	1	0	1	1	0	3	0.25	0.75
$X_3$	0	0	2	2	0	8	0.50	0.50
$X_4$	3	0	0	3	1	19	0.75	0.25

mutation operator tends to increase the diversity of the population. More details about the two operators can be found in [7] and in the Matlab code [8].

---

**Algorithm 1** Habitat migration

---

```

1: for  $i = 1$  to  $NP$  do
2:   Select  $X_i$  with probability  $\propto \lambda_i$ 
3:   if  $\text{rndreal}(0, 1) < \lambda_i$  then
4:     for  $j = 1$  to  $NP$  do
5:       Select  $X_j$  with probability  $\propto \mu_j$ 
6:       if  $\text{rndreal}(0, 1) < \mu_j$  then
7:         Randomly select a variable  $\sigma$  from  $X_j$ 
8:         Replace the corresponding variable in  $X_i$  with  $\sigma$ 
9:       end if
10:    end for
11:   end if
12: end for

```

---



---

**Algorithm 2** Habitat mutation

---

```

1: for  $i = 1$  to  $NP$  do
2:   Compute the probability  $P_i$ 
3:   Select a variable  $X_i(j)$  with probability  $\propto P_i$ 
4:   if  $\text{rndreal}(0, 1) < m_i$  then
5:     Replace  $X_i(j)$  with a randomly generated variable from its range
6:   end if
7: end for

```

---

*Example:* We adopt the test function  $f(X) = \sum_{i=1}^5 X(i)^2$  to illustrate the migration operator of BBO. The test function would be minimized. Suppose the population size  $NP = 4$ . The current population is shown in Table 1. From Table 1 we can see that the solution  $X_1$  has the highest HSI, and hence it has the lowest  $\lambda$  value and highest  $\mu$  value. While the solution  $X_4$  is the worst one in the current population. So it has the highest  $\lambda$  and lowest  $\mu$ . Based on the migration operator shown in Algorithm 1,  $X_1$  would be kept unchanged. The other solutions, i.e.,  $X_2 - X_4$ , can accept some SIVs from other solutions (especially from the solutions with higher  $\mu$  values). After the migration operation, the current population might be changed as shown in Table 2. Table 2 indicates that the current population is improved after the migration operation.

From the description and analysis of BBO, we can see that the BBO method has a good exploitation ability, because the migration operator can efficiently share the information between solutions. However, it may lack the exploration ability, because of the random mutation of BBO as shown in Algorithm 2. In addition, in the original BBO method, the individual is encoded as integer values, which limits it to combine with the real recombination operators.

Table 2: The population after the migration operation.

	$X(1)$	$X(2)$	$X(3)$	$X(4)$	$X(5)$	fitness (HSI)
$X_1$	0	1	0	1	0	2
$X_2$	1	0	0	1	0	2
$X_3$	1	0	2	1	0	4
$X_4$	0	0	1	1	1	3

### 3. Neighborhood search operator

Neighborhood search operator is frequently used in EAs [9], [1], [10], [11]. Especially, the neighborhood search operator is the main operator in the Evolutionary Programming (EP) algorithm [1]. In EP, the new offspring are obtained by giving a perturbation to the original individual. This means all offspring for the next generation are generated in the neighborhood of current solutions. Thus EP is able to improve the quality of solutions through a neighborhood search strategy. There are many neighborhood search operators in EP, for example, Gaussian mutation [1], Cauchy mutation [1], Lévy mutation [11], exponential mutation [12],  $t$  mutation [13], and so on. In this section, we will briefly introduce Gaussian mutation, Cauchy mutation, and Lévy mutation, which will be used in this work. The reasons chosen these three neighborhood search operators are based on two considerations. First, the three operators are widely and successfully used in EP algorithm. Second, they can be easily used for the real-coded variables.

#### 3.1. Gaussian mutation

The formula for the probability density function of the Gaussian distribution [14] is

$$f_{\mu, \sigma^2}(x) = \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (5)$$

where  $\mu$  is the mean and  $\sigma^2$  is the variance. To indicate that a real-valued random variable  $Y$  is normally distributed with mean  $\mu$  and variance  $\sigma^2 \geq 0$ , we write

$$Y \sim N(\mu, \sigma^2)$$

Then the Gaussian mutation with  $\mu = 0$  and  $\sigma = 1$  can be described as

$$X'_i(j) = X_i(j) + N_j(0, 1) \quad (6)$$

where  $X_i(j)$  is the  $j$ -th decision variable of individual  $X_i$  and  $N_j(0, 1)$  indicates that the random number is generated anew for each value of  $j$ .

#### 3.2. Cauchy mutation

The Cauchy distribution [1], [14] has the probability density function

$$f_t(x) = \frac{1}{\pi} \frac{t}{t^2 + x^2} \quad (7)$$

where  $x \in R$  and  $t > 0$  is a scale parameter. To indicate that a real-valued random variable  $Y$  is Cauchy distributed with  $t > 0$ , we write

$$Y \sim \delta(t)$$

The Cauchy mutation [1] with  $t = 1$  can be described as

$$X'_i(j) = X_i(j) + \delta_j(1) \quad (8)$$

where  $\delta_j(1)$  indicates that the random number is generated anew for each value of  $j$ .

### 3.3. Lévy mutation

Lévy distribution [15], like the Gaussian distribution and the Cauchy distribution, is stable and has probability density function that is analytically expressible. The Lévy distribution can be formulized as

$$L_{\alpha,\gamma} = \frac{1}{\pi} \int_0^{\infty} e^{-\gamma q^\alpha} \cos(qy) dq \quad (9)$$

where  $y \in R$ ,  $\gamma > 0$  is the scaling factor, and  $0 < \alpha < 2$  controls the shape of the distribution. The Lévy mutation [11] with  $\gamma = 1$  and  $\alpha = 0.8$  can be described as

$$X'_i(j) = X_i(j) + L_j(0.8) \quad (10)$$

where  $L_j(0.8)$  indicates that the random number is generated anew for each value of  $j$ .

## 4. Real-coded BBO: RCBBO

Inspired by the ideas of real-coded EAs [16], [17] and the neighborhood search operators successfully used in EP [1], [11], in this study, we propose a real-coded BBO approach, called RCBBO, for the global optimization problems in the continuous domain. In RCBBO, each individual is represented by a  $D$ -dimensional real parameter vector. Moreover, the neighborhood search operator is integrated into RCBBO to enhance its exploration ability. The pseudocode of RCBBO is described in Algorithm 3, where  $t$  is the generation counter and  $NP$  is the population size.

---

### Algorithm 3 Real-coded BBO: RCBBO

---

- 1: Generate the initial population  $P$  randomly
  - 2: Evaluate the fitness (HSI) for each individual in  $P$
  - 3: Initialize the generation counter  $t = 1$
  - 4: **while** The halting criterion is not satisfied **do**
  - 5:   Sort the population from best to worst
  - 6:   For each individual, map the HSI to the number of species
  - 7:   Calculate the immigration rate  $\lambda_i$  and the emigration rate  $\mu_i$  for each individual  $X_i$
  - 8:   Modify the population with the migration operator shown in Algorithm 1
  - 9:   Update the probability for each individual
  - 10:   Mutate the population with the mutation operator, which will be shown in Algorithm 4
  - 11:   Evaluate the population
  - 12:    $t = t + 1$
  - 13: **end while**
- 

#### 4.1. Individual representation and initialization

In RCBBO, each individual  $X_i = (X_i(1), \dots, X_i(D))$  is a real-coded vector. The individual is initialized as

$$X_i(j) = l_j + \text{rndreal}(0, 1) \times (u_j - l_j) \quad (11)$$

where  $i = 1, \dots, NP$ ,  $j = 1, \dots, D$ ,  $u_j$  and  $l_j$  is the upper bound and lower bound of  $X_i(j)$ , respectively.

#### 4.2. Mutation with neighborhood search operator

In order to enhance the exploration ability of RCBBO, the neighborhood search operator is integrated into the mutation operator to replace the randomly generated variable shown in Algorithm 2. The modified mutation operator is described in Algorithm 4, where  $X_i(j)$  is the  $j$ -th decision variable of individual  $X_i$ . The neighborhood search operator in Algorithm 4 might be the Gaussian mutation, the Cauchy mutation, or the Lévy mutation mentioned in Section 3.

---

**Algorithm 4** Modified habitat mutation

---

```

1: for  $i = 1$  to  $NP$  do
2:   Compute the probability  $P_i$ 
3:   Select a variable  $X_i(j)$  with probability  $\propto P_i$ 
4:   if  $\text{rndreal}(0, 1) < m_i$  then
5:     Update  $X_i(j)$  with a neighborhood search operator
6:   end if
7: end for

```

---

#### 4.3. Handling the 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. This method is also used in [18].

$$X_i(j) = \begin{cases} 2 \times l_j - X_i(j) & \text{if } X_i(j) < l_j \\ 2 \times u_j - X_i(j) & \text{if } X_i(j) > u_j \end{cases} \quad (12)$$

## 5. Experimental results and analysis

To evaluate the performance of the proposed RCBBO algorithm, 23 benchmark functions chosen from [1] are employed. These functions have been widely used in the literature [19], [20]. The benchmark functions are given in Table 3, where  $D$  is the number of variables; “optimal” is the minimum value of the function, and  $S \subseteq R^D$ . A more detailed description of these functions can be found in [1], 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 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  $\text{random}(0,1)$  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 RCBBO, 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 following parameters unless a change is mentioned.

- Population size:  $NP = 100$  [1], [11], [19];
- Habitat modification probability = 1 [7];
- Mutation probability:  $m_{max} = 0.005$  [7];
- Maximum Number of Fitness Function Evaluations (Max\_NFFE): For f01, f06, f10, f12, and f13, Max\_NFFE = 150000; for f03 - f05, Max\_NFFE = 500000; for f02 and f11, Max\_NFFE = 200000; For f07 - f09, Max\_NFFE = 300000; for f14, f16 - f19, f21, and f22, Max\_NFFE = 10000; for f15, Max\_NFFE = 40000; and for f20, Max\_NFFE = 20000.

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. All the algorithms are implemented in standard C++. The source code can be obtained from the first author upon request.

In this work, in order to show the improvement of the exploration ability with the neighborhood search operator, three neighborhood search operators described in Section 3 are adopted. In the experiments, RCBBO-G means RCBBO with Gaussian mutation. RCBBO-C is RCBBO with Cauchy mutation. And RCBBO-L indicates RCBBO with Lévy mutation. The BBO method also uses the real-coded representation. However, the mutation of BBO is shown in Algorithm 2, which does not use the neighborhood search operator.

### 5.2. Unimodal functions

For unimodal functions, the average results of 50 independent runs are summarized in Table 4. Fig. 1 shows the progress of the mean best errors<sup>1</sup> found by the four algorithms over 50 runs for f01 - f06. It is obvious that RCBBO (RCBBO-G, RCBBO-C, and RCBBO-L) performs significantly better than BBO consistently for f01 - f06 (6 out of 7 functions). For function f07, the quartic function, all approaches can obtain similar results. Moreover, for functions f01 - f06, RCBBO converges faster than BBO due to its better exploration ability. In general, the results shown in Table 4 and Fig. 1 indicate that the neighborhood search operator can improve the exploration ability effectively for the unimodal functions.

### 5.3. Multimodal functions with many local minima

For multimodal functions with many local minima, they are often regarded as being difficult to optimize [1]. For these functions, the final results are much more important since they reflect the algorithm's ability to escape from the poor local optima and locate a good near-global optimum [19]. Table 5 summarizes the average results of 50 independent runs for functions f08 - f13. It is apparent that all RCBBO approaches perform significantly better than BBO in terms of the final results for all six functions. This is also clearly reflected by the *t*-test<sup>2</sup>. Fig. 2 shows the mean best error curves for these functions. It can be seen that RCBBO displays a faster convergence rate than BBO when they are run for a longer time. The reason might be that the neighborhood search operator can explore the search space more sufficiently, and hence it can escape the local minima and approach the near-global optimum.

### 5.4. Multimodal functions with a few local minima

For the functions with a few local minima, i.e., f14 - f23, the major difference compared with functions f08 - f13 is that functions f14 - f23 appear to be simpler than f08 - f13 due to their low dimensionalities and a smaller number of local minima [1]. The experimental results are given in Table 6 and Fig. 3 for these functions. It is interesting that quite different results have been observed for these functions compared with functions with many local minima (f08 - f14). For nine (f14 and f16 - f23) out of ten functions, there are not significant difference among the RCBBO approaches and BBO. Only for function f15, RCBBO with Gaussian mutation, RCBBO-C, is significant better than BBO in terms of the final result. For two functions (i.e., f14 and f23), BBO slightly outperforms the RCBBO approaches.

## 6. Conclusions and future work

In this paper, we extend the original BBO algorithm and propose a real-coded BBO method, where each individual (habitat) is represented by the real parameter vector. In order to improve the exploration ability of RCBBO, the neighborhood search operator is integrated into the habitat mutation. In addition, three neighborhood search operators (i.e., Gaussian mutation, Cauchy mutation, and Lévy mutation), which have been widely used in EP, are chosen to verify the improvement of the exploration ability.

To verify the performance of our RCBBO approach, 23 benchmark functions of a wide range dimensions and diverse complexities are selected from the literature. And the results are compared with BBO, which does not use the neighborhood search operator. Experimental results show that: Firstly, our proposed real-coded BBO approach can obtain a good performance to solve the continuous global optimization problems. Secondly, the neighborhood search operator is able to improve the exploration ability effectively, especially for the high-dimensional problems. It is worth noting that we do not compare the performance of the three selected mutation operators, because it is out of the goals of this paper.

Since the habitat migration operator described in Algorithm 1 has a good exploitation ability, it can be hybrid with other EAs to balance the exploration ability and the exploitation ability. Our future work will consist on hybridizing BBO with other EAs for the global optimization problems.

---

<sup>1</sup>The error of a solution  $X$  is defined as  $f(X) - f(X^*)$ , where  $X^*$  is the global optimum of the function.

<sup>2</sup>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 [21].

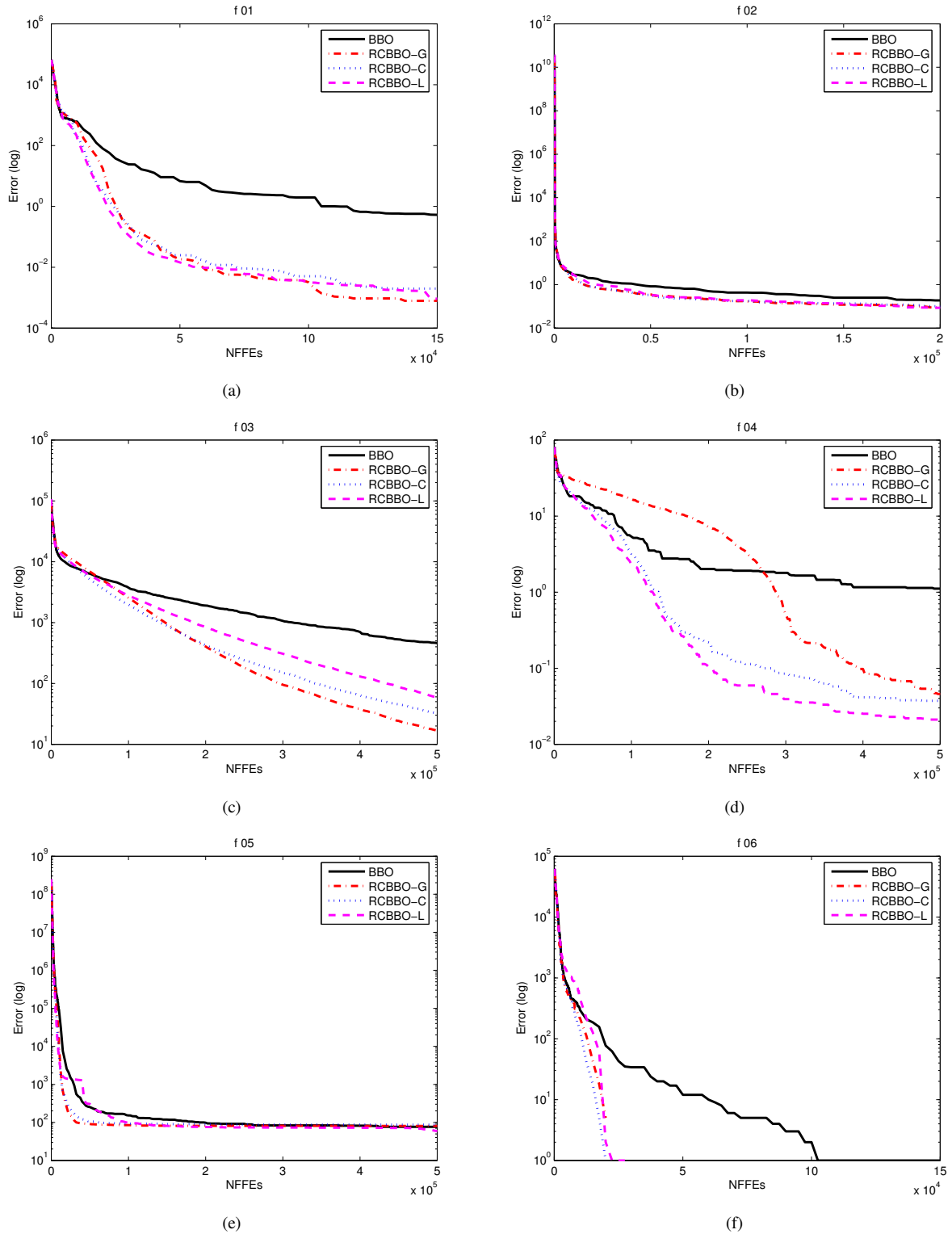


Figure 1: Mean best error curves of BBO, RCBBO-G, RCBBO-C, and RCBBO-L for six unimodal functions. (a) f01. (b) f02. (c) f03. (d) f04. (e) f05. (f) f06.

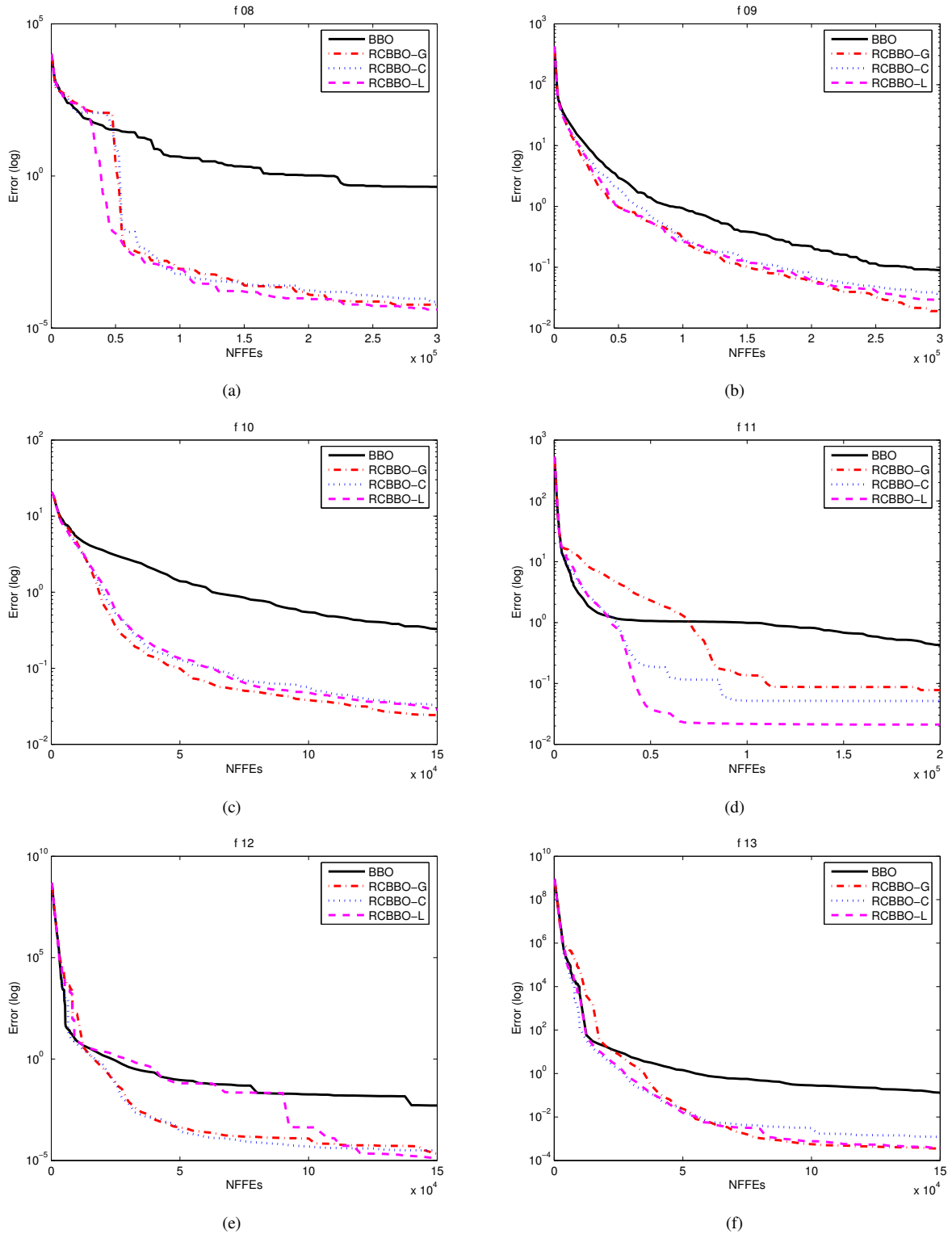


Figure 2: Mean best error curves of DE, BBO, RCBBO-G, RCBBO-C, and RCBBO-L for six multimodal functions with many local minima. (a) f08. (b) f09. (c) f10. (d) f11. (e) f12. (f) f13.

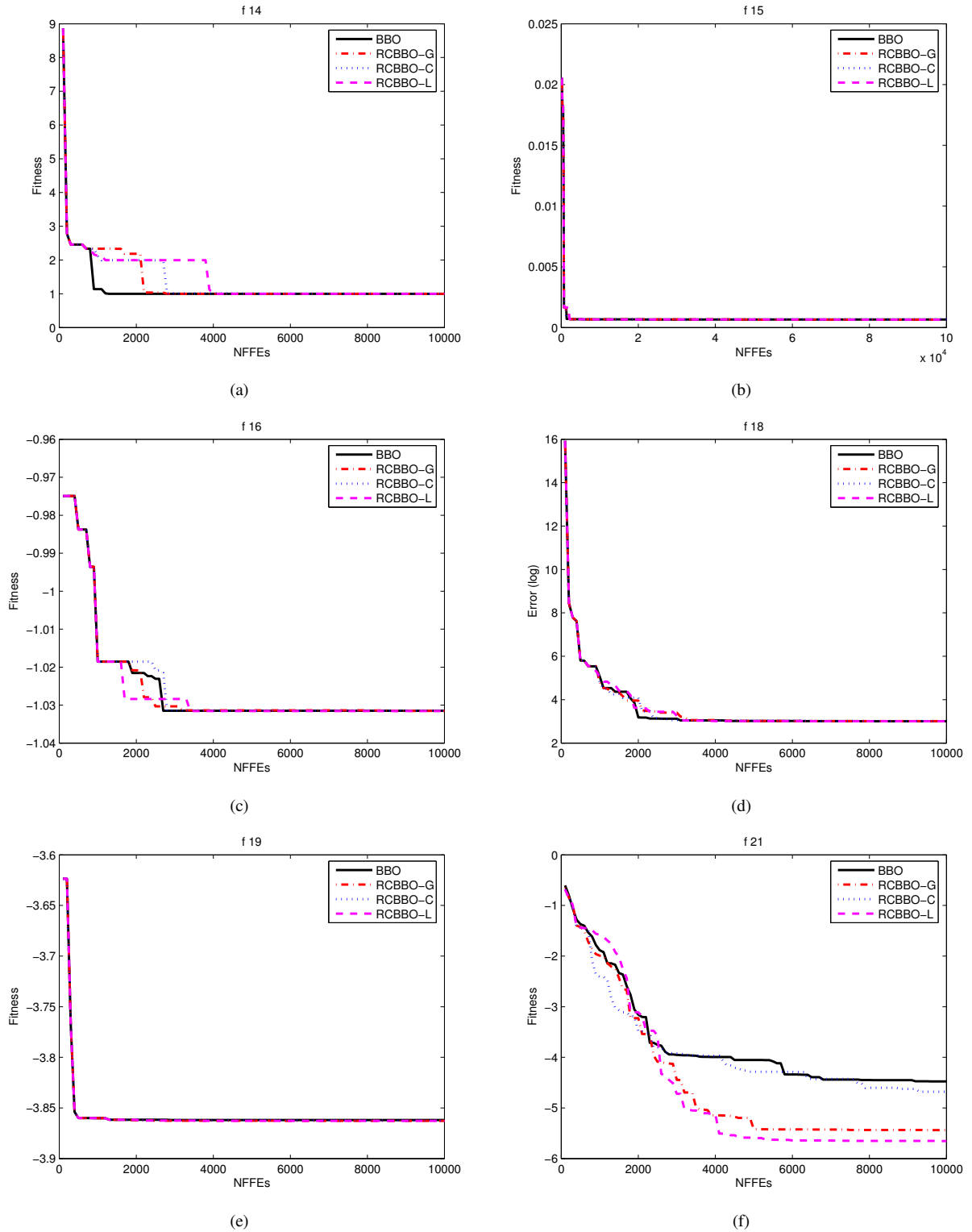


Figure 3: Mean best fitness curves of DE, BBO, RCBBO-G, RCBBO-C, and RCBBO-L for six multimodal functions with a few local minima. (a) f14. (b) f15. (c) f16. (d) f18. (e) f19. (f) f21.

## Acknowledgements

The authors sincerely thank Dr. D. Simon for his constructive comments on our work.

## References

- [1] X. Yao, Y. Liu, and G. Lin, Evolutionary programming made faster. *IEEE Transactions on Evolutionary Computation* 3: 82 - 102 (1999).
- [2] T. Bäck, U. Hammel, and H-P. Schwefel, Evolutionary computation: Comments on the history and current state. *IEEE Transactions on Evolutionary Computation* 1: 3 - 17 (1997).
- [3] K. Deep, M. Thakur. A new mutation operator for real coded genetic algorithms. *Applied Mathematics and Computation* 193: 211 - 230 (2007).
- [4] X. Yao and Y. Liu, Fast evolution strategies. *Control and Cybernetics* 26: 467 - 496 (1997).
- [5] J.J. Liang, A.K. Qin, P.N. Suganthan, et al, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Transactions on Evolutionary Computation* 10: 281 - 295 (2006).
- [6] R. Storn and K. Price, Differential evolution - A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization* 11: 341 - 359 (1997).
- [7] D. Simon, Biogeography-based optimization. *IEEE Transactions on Evolutionary Computation* 12: 702 - 713 (2008).
- [8] D. Simon, The Matlab code of biogeography-based optimization, (2008). Available online at: <http://academic.csuohio.edu/simond/bbo/>.
- [9] C.R. Reeves, Genetic algorithms and neighbourhood search. Selected papers from AISB workshop on Evolutionary Computing, LNCS 865: 115 - 130 (1994).
- [10] Z. Yang, J. He, and X. Yao, Making a Difference to Differential Evolution, in *Advances in Metaheuristics for Hard Optimization* (Z. Michalewicz and P. Siarry, Eds.), Natural Computing Series, Springer, 2007, p. 397 - 414.
- [11] C.Y. Lee and X. Yao, Evolutionary programming using mutations based on the Lévy probability distribution. *IEEE Transactions on Evolutionary Computation* 8: 1 - 13 (2004).
- [12] H. Narihisa, T. Taniguchi, M. Ohta, et al, Evolutionary programming with exponential mutation, in *Proceedings of Artificial Intelligence and Soft Computing* 1: 1 - 6 (2005).
- [13] W. Gong, Z. Cai, X. Lu, et al, A new mutation operator based on the T probability distribution in evolutionary programming, in *proceedings of the 5-th IEEE International Conference on Cognitive Informatics (ICCI2006)* 1: 675 - 679 (2006).
- [14] W. Feller, *An introduction to probability theory and its applications*, Vol. 2, 3rd ed. New York: Wiley, 1971.
- [15] B. Gnedenko and A. Kolmogorov, *Limit distribution for sums of independent random variables*. Cambridge, MA: Addition-Wesley, 1954.
- [16] I. Ono, H. Kita, and S. Kobayashi, A real-coded genetic algorithm using the unimodal normal distribution crossover, in *Advances in Evolutionary Computing: Theory and Applications* (A. Ghosh and S. Tsutsui, Eds.), Natural Computing Series, Springer, 2003, p. 213 - 237.
- [17] K. Deb, J. Dhiraj, and A. Ashish, Real coded evolutionary algorithms with parent centric recombination. in *Proceedings of the 2002 Congress on Evolutionary Computation* 1: 61 - 66 (2002).
- [18] J. Röykköen, S. Kukkonen, and K. Price, Real-parameter optimization with differential evolution, in *proceedings of the 2005 IEEE congress on evolutionary computation CEC2005* 1: 506 - 513 (2005).
- [19] J. Brest, S. Greiner, B. Bošković, et al. Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. *IEEE Transactions on Evolutionary Computation* 10: 646 - 657 (2006).
- [20] W. Gong, Z. Cai, and L. Jiang, Enhancing the performance of differential evolution using orthogonal design method, *Applied Mathematics and Computation* 206: 56 - 69 (2008).
- [21] C.H. Goulden, *Methods of Statistical Analysis, 2nd ed.* New York: Wiley, 1956, p. 50 - 55.

Table 3: Benchmark functions used in our experimental study. More details of all functions can be found in [1].

Test Functions	$D$	$S$	optimal
$f_{01} = \sum_{i=1}^n x_i^2$	30	$[-100, 100]^n$	0
$f_{02} = \sum_{i=1}^n  x_i  + \prod_{i=1}^n  x_i $	30	$[-10, 10]^n$	0
$f_{03} = \sum_{i=1}^n (\sum_{j=1}^i x_j)^2$	30	$[-100, 100]^n$	0
$f_{04} = \max\{ x_i , 1 \leq i \leq n\}$	30	$[-100, 100]^n$	0
$f_{05} = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30	$[-30, 30]^n$	0
$f_{06} = \sum_{i=1}^{n-1} (x_i + 0.5)^2$	30	$[-100, 100]^n$	0
$f_{07} = \sum_{i=1}^n x_i^4 + \text{random}[0, 1)$	30	$[-1.28, 1.28]^n$	0
$f_{08} = \sum_{i=1}^n (-x_i \sin(\sqrt{ x_i }))$	30	$[-500, 500]^n$	-12569.48662
$f_{09} = \sum_{i=1}^n (x_i^2 - 10 \cos(2\pi x_i) + 10)$	30	$[-5.12, 5.12]^n$	0
$f_{10} = -20 \exp(-0.2 \sqrt{\frac{1}{n} \sum_{i=1}^n x_i^2}) - \exp(\frac{1}{n} \sum_{i=1}^n \cos(2\pi x_i)) + 20 + \exp(1)$	30	$[-32, 32]^n$	0
$f_{11} = \frac{1}{4000} \sum_{i=1}^n x_i^2 - \prod_{i=1}^n \cos(\frac{x_i}{\sqrt{i}}) + 1$	30	$[-600, 600]^n$	0
$f_{12} = \frac{\pi}{n} (10 \sin^2(\pi y_i) + \sum_{i=1}^{n-1} (y_i - 1)^2 \cdot [1 + 10 \sin^2(\pi y_{i+1})]) + (y_n - 1)^2 + \sum_{i=1}^n u(x_i, 10, 100, 4)$	30	$[-50, 50]^n$	0
$f_{13} = \frac{1}{10} \{\sin^2(3\pi x_1) + \sum_{i=1}^{n-1} (x_i - 1)^2 [1 + \sin^2(3\pi x_{i+1})] + (x_n - 1)^2 [1 + \sin^2(2\pi x_n)]\} + \sum_{i=1}^n u(x_i, 5, 100, 4)$	30	$[-50, 50]^n$	0
$f_{14} = \left[ \frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^j (x_i - a_{ij})^6} \right]^{-1}$	2	$[-65.536, 65.536]^n$	0.998
$f_{15} = \sum_{i=1}^{11} \left[ a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4} \right]^2$	4	$[-5, 5]^n$	0.003075
$f_{16} = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	$[-5, 5]^n$	-1.0316285
$f_{17} = (x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})\cos x_1 + 10$	2	$[-5, 10] \times [0, 15]$	0.398
$f_{18} = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	2	$[0, 1]^n$	3
$f_{19} = -\sum_{i=1}^4 c_i \exp[-\sum_{j=1}^n a_{ij}(x_j - p_{ij})^2]$	3	$[0, 1]^n$	-3.86
$f_{20} = -\sum_{i=1}^4 c_i \exp[-\sum_{j=1}^n a_{ij}(x_j - p_{ij})^2]$	6	$[0, 1]^n$	-3.32
$f_{21} = -\sum_{i=1}^5 [(x - a_i)(x - a_i)^T + c_i]^{-1}$	4	$[0, 10]^n$	-10.1532
$f_{22} = -\sum_{i=1}^7 [(x - a_i)(x - a_i)^T + c_i]^{-1}$	4	$[0, 10]^n$	-10.4029
$f_{23} = -\sum_{i=1}^{10} [(x - a_i)(x - a_i)^T + c_i]^{-1}$	4	$[0, 10]^n$	-10.5364

## A real-coded BBO with neighborhood search operator

Table 4: Comparison of the experimental results, averaged over 50 independent runs, of BBO, RCBBO-G, RCBBO-C, and RCBBO-L for the unimodal functions (f01 - f07). “Mean” indicates the mean best error values found in the last generation, “Std Dev” stands for the standard deviation.  $t$ -test tests BBO against other algorithms, respectively. Hereafter, a result with **Boldface** means better value found.

$F$	$D$	Max_NFFEs	BBO Mean (Std Dev)	RCBBO-G Mean (Std Dev)	RCBBO-C Mean (Std Dev)	RCBBO-L Mean (Std Dev)
f01	30	150 000	8.86E-01 (3.26E-01)	<b>1.39E-03 (5.50E-04)</b> <sup>†</sup>	2.11E-03 (7.41E-04) <sup>†</sup>	1.63E-03 (6.60E-04) <sup>†</sup>
f02	30	200 000	2.42E-01 (4.58E-02)	<b>7.99E-02 (1.44E-02)</b> <sup>†</sup>	9.15E-02 (1.51E-02) <sup>†</sup>	8.04E-02 (1.42E-02) <sup>†</sup>
f03	30	500 000	4.16E+02 (2.02E+02)	<b>2.27E+01 (1.03E+01)</b> <sup>†</sup>	3.90E+01 (1.91E+01) <sup>†</sup>	4.80E+01 (2.19E+01) <sup>†</sup>
f04	30	500 000	7.76E-01 (1.72E-01)	3.09E-02 (7.27E-03) <sup>†</sup>	3.02E-02 (5.29E-03) <sup>†</sup>	<b>2.68E-02 (5.09E-03)</b> <sup>†</sup>
f05	30	500 000	9.14E+01 (3.78E+01)	5.54E+01 (3.52E+01) <sup>†</sup>	6.45E+01 (3.43E+01) <sup>†</sup>	<b>5.27E+01 (3.91E+01)</b> <sup>†</sup>
f06	30	150 000	2.80E-01 (5.36E-01)	<b>0.00E+00 (0.00E+00)</b> <sup>†</sup>	<b>0.00E+00 (0.00E+00)</b> <sup>†</sup>	<b>0.00E+00 (0.00E+00)</b> <sup>†</sup>
f07	30	300 000	1.90E-02 (7.29E-03)	<b>1.75E-02 (6.43E-03)</b>	1.95E-02 (6.96E-03)	1.87E-02 (5.11E-03)

<sup>†</sup> The value of  $t$  with 49 degrees of freedom is significant at  $\alpha = 0.05$  by two-tailed test.

Table 5: Comparison of the experimental results for the multimodal functions with many local minima (f08 - f13).

$F$	$D$	Max_NFFEs	BBO Mean (Std Dev)	RCBBO-G Mean (Std Dev)	RCBBO-C Mean (Std Dev)	RCBBO-L Mean (Std Dev)
f08	30	300 000	-12569.0 (1.65E-01)	<b>-12569.5 (2.20E-05)</b> <sup>†</sup>	<b>-12569.5 (2.65E-05)</b> <sup>†</sup>	<b>-12569.5 (1.93E-05)</b> <sup>†</sup>
f09	30	300 000	8.50E-02 (3.42E-02)	<b>2.62E-02 (9.76E-03)</b> <sup>†</sup>	3.39E-02 (1.51E-02) <sup>†</sup>	2.77E-02 (1.02E-02) <sup>†</sup>
f10	30	150 000	3.48E-01 (7.06E-02)	<b>2.51E-02 (5.51E-03)</b> <sup>†</sup>	3.34E-02 (6.15E-03) <sup>†</sup>	2.89E-02 (5.15E-03) <sup>†</sup>
f11	30	300 000	4.82E-01 (1.27E-01)	8.49E-02 (5.44E-02) <sup>†</sup>	3.57E-02 (3.78E-02) <sup>†</sup>	<b>2.99E-02 (3.20E-02)</b> <sup>†</sup>
f12	30	150 000	5.29E-03 (5.21E-03)	3.28E-05 (3.33E-05) <sup>†</sup>	5.21E-05 (5.69E-05) <sup>†</sup>	<b>2.73E-05 (2.49E-05)</b> <sup>†</sup>
f13	30	150 000	1.42E-01 (5.14E-02)	<b>3.72E-04 (4.63E-04)</b> <sup>†</sup>	6.96E-04 (1.02E-03) <sup>†</sup>	5.84E-04 (8.82E-04) <sup>†</sup>

<sup>†</sup> The value of  $t$  with 49 degrees of freedom is significant at  $\alpha = 0.05$  by two-tailed test.

Table 6: Comparison of the experimental results for the multimodal functions with a few local minima (f14 - f23).

$F$	$D$	Max_NFFEs	BBO Mean (Std Dev)	RCBBO-G Mean (Std Dev)	RCBBO-C Mean (Std Dev)	RCBBO-L Mean (Std Dev)
f14	2	10 000	<b>0.998013 (2.74E-05)</b>	0.998017 (5.23E-05)	0.998086 (4.56E-04)	0.998069 (4.46E-04)
f15	4	100 000	9.00E-04 (2.68E-04)	<b>7.86E-04 (1.80E-04)</b> <sup>†</sup>	1.17E-03 (2.28E-03)	1.17E-03 (2.78E-03)
f16	2	10 000	-1.03095 (1.09E-03)	-1.03101 (9.01E-04)	-1.03110 (7.90E-04)	<b>-1.03112 (5.88E-04)</b>
f17	2	10 000	0.398327 (4.26E-04)	0.398414 (6.77E-04)	0.398470 (1.06E-03)	<b>0.398289 (5.52E-04)</b>
f18	2	10 000	3.007858 (9.57E-03)	3.009504 (1.12E-02)	3.008666 (1.15E-02)	<b>3.006942 (1.02E-02)</b>
f19	4	10 000	-3.86253 (2.62E-04)	-3.86248 (3.65E-04)	<b>-3.86254 (2.74E-04)</b>	-3.86247 (4.67E-04)
f20	6	20 000	-3.30741 (3.90E-02)	<b>-3.31691 (2.36E-02)</b>	-3.30748 (3.90E-02)	-3.31228 (3.26E-02)
f21	4	10 000	-4.49193 (3.34E+00)	-5.51341 (3.35E+00)	-4.61873 (3.32E+00)	<b>-5.61985 (3.45E+00)</b>
f22	4	10 000	-6.73583 (3.40E+00)	-6.80022 (3.52E+00)	-6.86903 (3.51E+00)	<b>-7.06758 (3.48E+00)</b>
f23	4	10 000	<b>-7.80261 (3.29E+00)</b>	-7.28480 (3.38E+00)	-7.25011 (3.47E+00)	-7.46472 (3.49E+00)

<sup>†</sup> The value of  $t$  with 49 degrees of freedom is significant at  $\alpha = 0.05$  by two-tailed test.