Author's personal copy

Applied Mathematics and Computation 216 (2010) 2749–2758

Contents lists available at ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

A real-coded biogeography-based optimization with mutation ☆

Wenyin Gong [a,*], Zhihua Cai [a], Charles X. Ling [b], Hui Li [a]

[a] School of Computer Science, China University of Geosciences, Wuhan 430074, PR China
[b] Department of Computer Science, The University of Western Ontario, London, Ontario, Canada N6A 5B7

ARTICLE INFO

Keywords:
Biogeography-based optimization
Mutation
Global optimization
Evolutionary programming
Exploration ability

ABSTRACT

Biogeography-based optimization (BBO) is a new biogeography inspired algorithm for global optimization. There are some open research questions that need to be addressed for BBO. In this paper, we extend the original BBO and present a real-coded BBO approach, referred to as RCBBO, for the global optimization problems in the continuous domain. Furthermore, in order to improve the diversity of the population and enhance the exploration ability of RCBBO, the mutation operator is integrated into RCBBO. 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. Moreover, experimental results also show that the mutation operator can improve the performance of RCBBO effectively.

Crown Copyright © 2010 Published by Elsevier Inc. All rights reserved.

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 $S \subseteq \mathbb{R}^D$ is a bounded set on $\mathbb{R}^D$ and $f : S \to \mathbb{R}$ 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

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

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

The major challenge of the global continuous 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 solving the global continuous optimization problems. There are many different EAs for global optimization in continuous domain, 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. The BBO approach has a way of sharing information between solutions. This feature is like other biology-based algorithms, such as GAs and PSO. However, BBO also has some features that are unique among biology-based algorithms, for example it maintains its set of solutions from one iteration to the next one, relying on migration to probabilistically adapt those solutions [7]. In the original BBO algorithm, it adopts a vector of integers to represent a solution to some problems. Simon compared BBO with seven

☆ This work was supported by the Fund for Outstanding Doctoral Dissertation of CUG, China Scholarship Council under Grant No. 2008641008, and the National High Technology Research and Development Program of China under Grant No. 2009AA12Z117.

* Corresponding author.
E-mail addresses: cug11100304@yahoo.com.cn (W. Gong), zhcai@cug.edu.cn (Z. Cai), cling@csd.uwo.ca (C.X. Ling), huili@vip.sina.com (H. Li).

0096-3003/$ - see front matter Crown Copyright © 2010 Published by Elsevier Inc. All rights reserved.
doi:10.1016/j.amc.2010.03.123

Author's personal copy

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 some open research questions that need to 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 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. Thus, it can utilize the information of current population efficiently. However, the migration operator lacks the exploration ability and cannot improve the diversity of the population.

The aims of this paper are twofold: (i) we propose an extension of the original BBO algorithm (RCBBO), where each individual is directly encoded by floating point for the global continuous optimization problems; and (ii) the mutation operator is integrated into RCBBO to enhance its exploration ability and to improve the diversity of the population. 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. Moreover, experimental results also show that the mutation operator can improve the performance of RCBBO effectively. Additionally, we combine the RCBBO with the selection rule of EP (RCBBO-EP) and compare it with some well-known EP variants. The results indicate that RCBBO-EP is better than, or highly competitively to, its competitors.

The remainder of this paper is organized as follows. In Section 2, the original BBO algorithm is briefly introduced. Some mutation 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 6, is devoted to the conclusions and future work.

2. Biogeography-based optimization: BBO

BBO [7] is a new biogeography inspired global optimization algorithm, which is similar to the island model-based GAs [8]. Each individual is considered as a "habitat" with a habitat suitability index (HSI) to measure the individual. The variables of the individual that characterize habitability are called suitability index variables (SIVs). In BBO, each individual has its own immigration rate λ and emigration rate μ. The immigration rate and 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; and 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 species models [7].

In BBO, there are two main operators, i.e., migration and mutation. Suppose that we have a global optimization problem and a population of candidate individuals. The individual is represented by a D-dimensional integer vector (SIV). The population consists of $NP = n$ parameter vectors X_i, $i = 1,\ldots, NP$. One option for implementing the migration operator and the mutation operator can be described in Algorithms 1 and 2, respectively. Where 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. m_i is the mutation rate that is calculated as:

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

where m_{max} is an user-defined parameter, and $P_{max} = \arg\max P_i$, $i = 1,\ldots, NP$. With the migration operator BBO can share the information between solutions. Additionally, the 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 [9].

Algorithm 1 (*Habitat migration*).

```
1: for i = 1 to NP
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 an SIV σ from X_j
8:             Replace a random SIV in X_i with σ
9:          end if
10:      end for
11:   end if
12: end for
```

Author's personal copy

Algorithm 2 (*Habitat mutation*).

```
1: for i = 1 to NP
2:    Compute the probability P_i
3:    Select SIV X_i(j) with probability ∝P_i
4:    if rndreal (0,1) < m_i then
5:       Replace X_i(j) with a randomly generated SIV
6:    end if
7: end for
```

According to the introduction of BBO, we can see that the migration operator is able to 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.

3. Mutation operator

Mutation operator is frequently used in EAs [1,10–12]. Especially, the mutation 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 mutation strategy. There are many mutation operators in EP, for example, Gaussian mutation [1], Cauchy mutation [1], Lévy mutation [12], exponential mutation [13], t mutation [14], mixed mutation strategy [15], 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 for choosing these three mutation operators are based on three considerations: First, the three operators are widely and successfully used in EP. Second, they can be easily used for the real-coded variables. Third, the mutation operators of EP is capable of performing either local search or global search depending on the step size; especially in the early and middle stages of the evolution, the mutation operators can enhance the exploration and improve the diversity of the population.

3.1. Gaussian mutation

The formula for the probability density function of the Gaussian distribution [16] is

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

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

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

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

$$X_i'(j) = X_i(j) + N_j(0, 1), \tag{6}$$

where $X_i(j)$ is the jth 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 [1,16] has the probability density function

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

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

$$X_i'(j) = X_i(j) + \delta_j(1), \tag{8}$$

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

3.3. Lévy mutation

Lévy distribution [17], 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

Author's personal copy

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

where $y \in R$, $\gamma > 0$ is the scaling factor, and $0 < \alpha < 2$ controls the shape of the distribution. The Lévy mutation [12] with $\gamma = 1$ and $\alpha = 0.8$ can be described as

$$X_i'(j) = X_i(j) + L_j(0.8), \tag{10}$$

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 [18,19] and the mutation operators successfully used in EP [1,12], 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 mutation operator is integrated into RCBBO to enhance its exploration ability and to improve the diversity of the population. 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 λ_i and the emigration rate μ_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 $X_i = (X_i(1),\ldots,X_i(D))$ is a real-coded vector. The individual is initialized as

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

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

4.2. Mutation with mutation operator

In order to enhance the exploration ability of BBO, the mutation operator is integrated into BBO to replace the randomly generated SIV shown in Algorithm 2. The modified mutation operator is described in Algorithm 4, where $X_i(j)$ is the jth decision variable of individual X_i. The mutation 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 SIV $X_i(j)$ with probability $\propto P_i$
4: **if** rndreal $(0,1) < m_i$ **then**
5: Update $X_i(j)$ with a mutation operator
6: **end if**
7: **end for**

Author's personal copy

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 [20].

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

4.4. Differences among our approach, BBO, and EP

With regard to the differences among our approach, the original BBO algorithm, and the EP algorithm, we can see that:

- Compared with BBO, there are two main differences: (i) Our approach is encoded by the floating point, while BBO is encoded by the integer point; (ii) since the floating point is used in our approach, the real mutation operator in EAs can be easily adopted in RCBBO, and hence, the exploration ability can be enhanced.
- Compare with EP, the migration operator is used in RCBBO, it can utilize the population information effectively. In addition, the modified habitat mutation shown in Algorithm 4 can balance the exploration and the exploitation of RCBBO.

5. Experimental results and analysis

To evaluate the performance of the proposed RCBBO algorithm, 23 benchmark functions from [1] are employed. These functions have be widely used in the literature [21,22]. The benchmark functions are given in Table 1, 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. In addition, four rotated functions (F03, F07, F08, and F10) are chosen from CEC'05 test functions [23].

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

Test functions	n	S	Optimal
$f_{01} = \sum_{i=1}^{n} x_i^2$	30	$[-100,100]^n$	0
$f_{02} = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	$[-10,10]^n$	0
$f_{03} = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2$	30	$[-100,100]^n$	0
$f_{04} = \max_i \{ \lvert x_i \rvert, 1 \leqslant i \leqslant n \}$	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
$f_{06} = \sum_{i=1}^{n-1} (\lfloor x_i + 0.5 \rfloor)^2$	30	$[-100,100]^n$	0
$f_{07} = \sum_{i=1}^{n} x_i^4 + random[0,1)$	30	$[-1.28,1.28]^n$	0
$f_{08} = \sum_{i=1}^{n} \left(-x_i \sin\left(\sqrt{\lvert x_i \rvert} \right) \right)$	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
$f_{10} = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2} \right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi x_i) \right) + 20 + \exp(1)$	30	$[-32,32]^n$	0
$f_{11} = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}} \right) + 1$	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
$f_{14} = \left[\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} (x_i - a_{ij})^6} \right]^{-1}$	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
$f_{16} = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1 x_2 - 4x_2^2 + 4x_2^4$	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
$f_{18} = [1 + (x_1 + x_2 + 1)^2 (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1 x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2)^2 (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1 x_2 + 27x_2^2)]$	2	$[0,1]^n$	3
$f_{19} = -\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{n} a_{ij} (x_j - p_{ij})^2 \right]$	3	$[0,1]^n$	−3.86
$f_{20} = -\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{n} a_{ij} (x_j - p_{ij})^2 \right]$	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

Author's personal copy

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 *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: $NP = 100$;
- habitat modification probability = 1;
- mutation probability: $m_{max} = 0.005$;
- maximum Number of Fitness Function Evaluations (Max_NFFEs): For f01, f06, f10, f12, and f13, Max_NFFEs = 150,000; for f03–f05, Max_NFFEs = 500,000; for f02 and f11, Max_NFFEs = 200,000; For f07–f09, Max_NFFEs = 300,000; for f14, f16–f19, f21, and f22, Max_NFFEs = 10,000; for f15, Max_NFFEs = 40,000; and for f20, Max_NFFEs = 20,000.

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 mutation operator, three mutation 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.

5.2. Unimodal functions

For unimodal functions, the average results of 50 independent runs are summarized in Table 2. Fig. 1 shows the progress of the mean best errors[1] found by the four algorithms over 50 runs for the selected functions. 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, from Fig. 1 we can see that RCBBO converges faster than BBO due to its better exploration ability. The results shown in Table 2 and Fig. 1 indicate that the mutation 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 [21]. Table 2 summarizes the average results of 50 independent runs for the selected functions. 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.[2] Fig. 1 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 mutation operator can explore the search space and improve the diversity of the population 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 2 and Fig. 1 for functions f14–f23. It is interesting that quite different results have be observed for these functions compared with functions with many local minima (f08–f14). For nine (f14 and f16–f23) out of 10 functions, there are not significant difference between 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.

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

[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 [24].

Author's personal copy

Table 2
Comparison of the experimental results, averaged over 50 independent runs, of BBO, RCBBO-G, RCBBO-C, and RCBBO-L for all of the test functions. "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)	**1.39E−03 (5.50E−04)**[a]	2.11E−03 (7.41E−04)[a]	1.63E−03 (6.60E−04)[a]
f02	30	200,000	2.42E−01 (4.58E−02)	**7.99E−02 (1.44E−02)**[a]	9.15E−02 (1.51E−02)[a]	8.04E−02 (1.42E−02)[a]
f03	30	500,000	4.16E+02 (2.02E+02)	**2.27E+01 (1.03E+01)**[a]	3.90E+01 (1.91E+01)[a]	4.80E+01 (2.19E+01)[a]
f04	30	500,000	7.76E−01 (1.72E−01)	3.09E−02 (7.27E−03)[a]	3.02E−02 (5.29E−03)[a]	**2.68E−02 (5.09E−03)**[a]
f05	30	500,000	9.14E+01 (3.78E+01)	5.54E+01 (3.52E+01)[a]	6.45E+01 (3.43E+01)[a]	**5.27E+01 (3.91E+01)**[a]
f06	30	150,000	2.80E−01 (5.36E−01)	**0.00E+00 (0.00E+00)**[a]	**0.00E+00 (0.00E+00)**[a]	**0.00E+00 (0.00E+00)**[a]
f07	30	300,000	1.90E−02 (7.29E−03)	**1.75E−02 (6.43E−03)**	1.95E−02 (6.96E−03)	1.87E−02 (5.11E−03)
f08	30	300,000	−12569.0 (1.65E−01)	**−12569.5 (2.20E−05)**[a]	**−12569.5 (2.65E−05)**[a]	**−12569.5 (1.93E−05)**[a]
f09	30	300,000	8.50E−02 (3.42E−02)	**2.62E−02 (9.76E−03)**[a]	3.39E−02 (1.51E−02)[a]	2.77E−02 (1.02E−02)[a]
f10	30	150,000	3.48E−01 (7.06E−02)	**2.51E−02 (5.51E−03)**[a]	3.34E−02 (6.15E−03)[a]	2.89E−02 (5.15E−03)[a]
f11	30	300,000	4.82E−01 (1.27E−01)	8.49E−02 (5.44E−02)[a]	3.57E−02 (3.78E−02)[a]	**2.99E−02 (3.20E−02)**[a]
f12	30	150,000	5.29E−03 (5.21E−03)	3.28E−05 (3.33E−05)[a]	5.21E−05 (5.69E−05)[a]	**2.73E−05 (2.49E−05)**[a]
f13	30	150,000	1.42E−01 (5.14E−02)	**3.72E−04 (4.63E−04)**[a]	6.96E−04 (1.02E−03)[a]	5.84E−04 (8.82E−04)[a]
f14	2	10,000	**0.998013 (2.74E−05)**	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)	**7.86E−04 (1.80E−04)**[a]	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)	**−1.03112 (5.88E−04)**
f17	2	10,000	0.398327 (4.26E−04)	0.398414 (6.77E−04)	0.398470 (1.06E−03)	**0.398289 (5.52E−04)**
f18	2	10,000	3.007858 (9.57E−03)	3.009504 (1.12E−02)	3.008666 (1.15E−02)	**3.006942 (1.02E−02)**
f19	4	10,000	−3.86253 (2.62E−04)	−3.86248 (3.65E−04)	**−3.86254 (2.74E−04)**	−3.86247 (4.67E−04)
f20	6	20,000	−3.30741 (3.90E−02)	**−3.31691 (2.36E−02)**	−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)	**−5.61985 (3.45E+00)**
f22	4	10,000	−6.73583 (3.40E+00)	−6.80022 (3.52E+00)	−6.86903 (3.51E+00)	**−7.06758 (3.48E+00)**
f23	4	10,000	**−7.80261 (3.29E+00)**	−7.28480 (3.38E+00)	−7.25011 (3.47E+00)	−7.46472 (3.49E+00)
F03	10	100,000	6.19E+05 (4.31E+05)	5.30E+05 (4.20E+05)[a]	5.41E+05 (2.83E+05)[a]	**2.97E+05 (1.19E+05)**[a]
F07	10	100,000	1.70E+00 (8.86E−01)	1.00E+00 (6.41E−01)[a]	9.25E−01 (1.32E−01)[a]	**5.88E−01 (3.44E−01)**[a]
F08	10	100,000	2.04E+01 (6.65E−02)	2.03E+01 (9.71E−02)	2.03E+01 (9.11E−02)	2.03E+01 (1.08E−01)
F10	10	100,000	9.81E+00 (1.77E+00)	**5.91E+00 (1.87E+00)**[a]	7.04E+00 (1.85E+00)[a]	6.02E+00 (2.35E+00)[a]

[a] The value of *t* with 49 degrees of freedom is significant at $\alpha = 0.05$ by two-tailed test.

5.5. Rotated functions

With respect to the rotated functions (F03, F07, F08, and F10), all of these functions are tested at $D = 10$ with Max_NFFEs = 100,000. The results are shown in Table 2. From Table 2, it can be seen that BBO is significantly outperformed by RCBBO-G, RCBBO-C, and RCBBO-L on three out of four functions. For function F08, all of the four algorithms obtains the similar results.

5.6. Comparison with EP variants

Since the BBO migration operator is combined with the mutation operator mainly used in the EP algorithm, in this section, we compare our approach with some EP variants, i.e., MSEP [15], FEP [1], and CEP [1]. To make a fair comparison with these EP methods, RCBBO shown in Algorithm 3 is combined with the EP selection rule, hence, our approach is referred to as RCBBO with EP selection (RCBBO-EP). All parameters are kept unchanged as described in Section 5.1. In RCBBO-EP, the Gaussian mutation is used as the illustration. The tournament size $q = 10$ for selection as used in [1,15]. The Max_NFFEs for each function are used as listed in the third columns of Table 2, only except for function f15, where Max_NFFEs = 400,000 [1,15]. The results are shown in Table 3. For MSEP, the results are obtained from [15], except for f05, we omitted the result, since the Max_NFFEs = 150,000 are used in [15]. For FEP and CEP, the results are obtained from [1]. Note that for functions f05, f08, and f09, the Max_NFFEs used in RCBBO-EP are less than those of FEP and CEP.

With respect to MSEP, RCBBO-EP is significantly better than MSEP on 10 out of 21 functions. For functions f21, f22, and f23, RCBBO-EP is outperformed by MSEP. RCBBO-EP gets stuck in the local minima in several runs for these functions. The reason might be that the Gaussian mutation in RCBBO-EP is still suffered by these problems like CEP. The problem may be remedied using the mixed mutation strategy as proposed in MSEP. For the rest 8 functions, there are no significant differences between these two algorithms. From Table 3, we can also see that on the majority of the high-dimensional problems, RCBBO-EP is significantly better than MSEP (9 out of 10). This is confirmed the efficient population utilization of the migration operator of BBO.

Compared with FEP and CEP, on 21 out of 23 functions RCBBO-EP is significantly better than FEP. Only for two functions f06 and f19, there are no significant differences between RCBBO-EP and FEP. The similar conclusion can be drawn about the results between RCBBO-EP and CEP, RCBBO-EP is significantly better than CEP on the majority of the functions (21 out of 23).

Author's personal copy

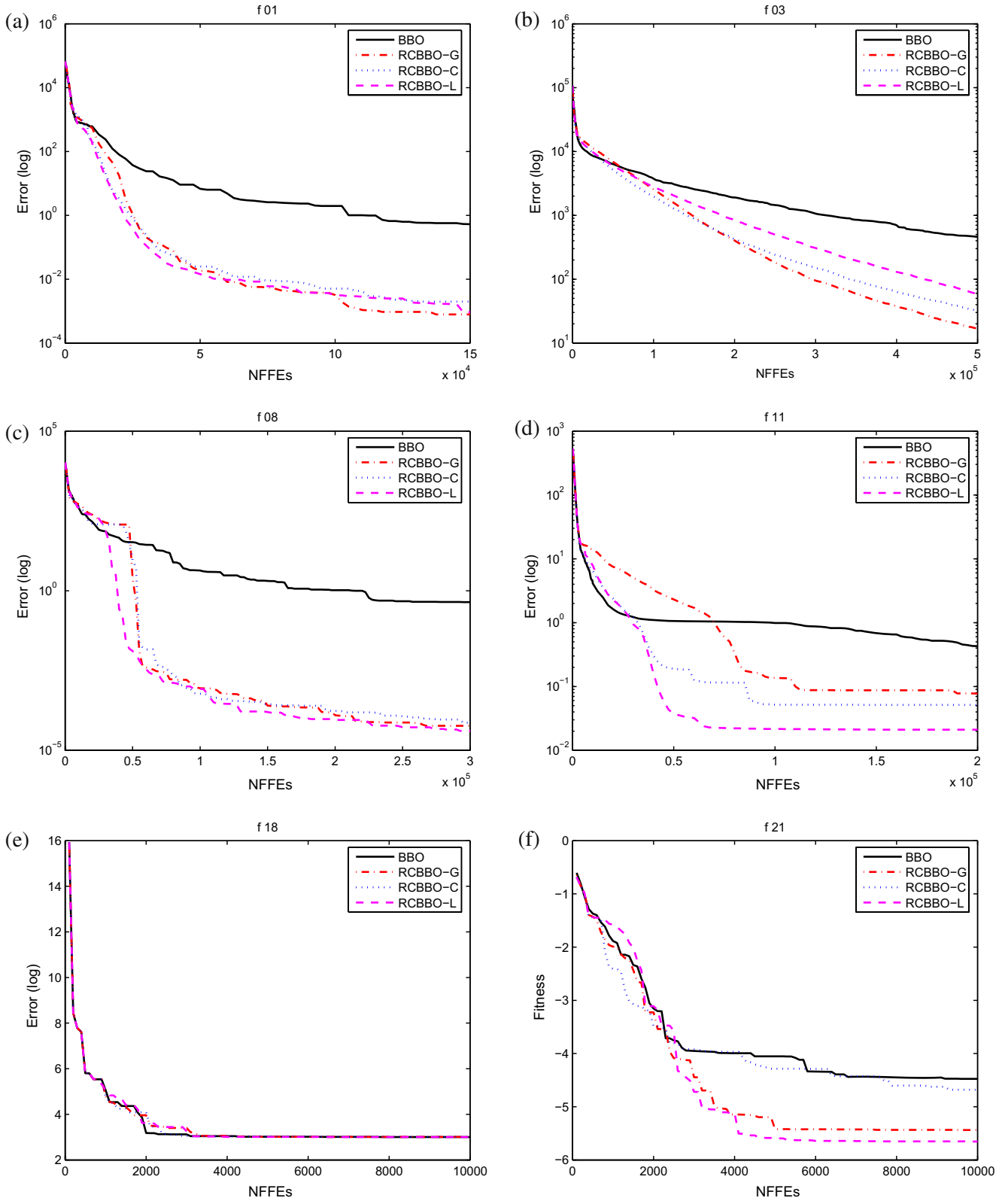

Fig. 1. Mean best error curves of BBO, RCBBO-G, RCBBO-C, and RCBBO-L for the selected functions. (a) f01. (b) f03. (c) f08. (d) f11. (e) f18. (f) f21.

In summary, when the migration operator of BBO is combined with the mutation operator and the selection rule of EP, RCBBO-EP is better than, or highly competitively to, MSEP, FEP, and CEP on the test functions. The migration operator is able to efficiently utilize the information of current population. The mutation operator can improve the diversity and enhance the exploration in the early and middle stages of the evolution. In addition, the selection rule of EP can also improve the performance of RCBBO when comparing the results between Tables 3 and 2.

Author's personal copy

Table 3
Comparison of the experimental results between RCBBO-EP with EP variants on all test functions. "NA" means not available.

F	RCBBO-EP	MSEP [15]	FEP [1]	CEP [1]
f01	**2.35E−06 (3.17E−07)**	1.00E−04 (1.30E−05)[a]	5.70E−04 (1.30E−04)[a]	2.20E−04 (5.90E−04)[a]
f02	**5.03E−07 (3.17E−08)**	4.10E−04 (2.10E−04)[a]	8.10E−03 (7.70E−04)[a]	2.60E−03 (1.70E−04)[a]
f03	**1.59E−03 (2.90E−03)**	NA (NA)	1.60E−02 (1.40E−02)[a]	5.00E−02 (6.60E−02)[a]
f04	**6.66E−04 (4.62E−05)**	2.70E−02 (1.70E−02)[a]	3.00E−01 (5.00E−01)[a]	2.00E+00 (1.20E+00)[a]
f05	**2.42E+00 (2.60E+00)**	NA (NA)	5.06E+00 (5.87E+00)[a]	6.17E+00 (1.36E+01)[a]
f06	0 (0.00E+00)	0 (0.00E+00)	0 (0.00E+00)	577.76 (1.13E+03)[a]
f07	**1.70E−03 (5.90E−04)**	6.10E−03 (1.70E−03)[a]	7.60E−03 (2.60E−03)[a]	1.80E−02 (6.40E−03)[a]
f08	−12569.5 (2.65E−06)	−12569.5 (3.40E−04)	−12554.5 (5.26E+01)[a]	−7917.1 (6.35E+02)[a]
f09	**3.97E−06 (5.32E−06)**	2.50E−05 (2.10E−05)[a]	4.60E−02 (1.20E−02)[a]	8.90E+01 (2.31E+01)[a]
f10	**9.40E−04 (5.84E−05)**	1.70E−03 (4.30E−04)[a]	1.80E−02 (2.10E−03)[a]	9.20E+00 (2.80E+00)[a]
f11	**6.70E−08 (9.55E−09)**	8.50E−04 (2.30E−03)[a]	1.60E−02 (2.20E−02)[a]	8.60E−02 (1.20E−01)[a]
f12	**1.57E−08 (2.40E−09)**	7.50E−07 (4.00E−07)[a]	9.20E−06 (3.60E−06)[a]	1.76E+00 (2.40E+00)[a]
f13	**2.52E−07 (1.01E−07)**	1.20E−05 (1.10E−05)[a]	1.60E−04 (7.30E−05)[a]	1.40E+00 (3.70E+00)[a]
f14	0.998004 (0.00E+00)	0.998004 (0.00E+00)	1.22 (5.60E−01)[a]	1.66 (1.19E+00)[a]
f15	**0.000308 (1.49E−09)**	3.08E−04 (1.30E−06)[a]	0.0005 (3.20E−04)[a]	0.00047 (3.00E−04)[a]
f16	−1.03163 (7.89E−11)	−1.03 (0.00E+00)	−1.03 (4.90E−04)[a]	−1.03 (4.90E−04)[a]
f17	0.397887 (3.26E−11)	0.398 (0.00E+00)	0.398 (1.50E−07)[a]	0.398 (1.50E−07)[a]
f18	3 (4.58E−09)	3 (0.00E+00)	3.02 (1.10E−01)[a]	3 (0.00E+00)
f19	−3.86277 (6.89E−05)	−3.86 (0.00E+00)	−3.86 (1.40E−05)	−3.86 (1.40E−02)
f20	**−3.322 (0.00E+00)**	−3.32 (1.70E−04)	−3.27 (5.90E−02)[a]	−3.28 (5.80E−02)[a]
f21	−9.85005 (1.21E+00)	**−10.15 (5.00E−05)**[b]	−5.52 (1.59E+00)[a]	−6.86 (2.67E+00)[a]
f22	−10.2975 (7.46E−01)	**−10.4 (4.70E−06)**[b]	−5.52 (2.12E+00)[a]	−8.27 (2.95E+00)[a]
f23	−10.3484 (9.31E−01)	**−10.54 (1.30E−04)**[b]	−6.57 (3.14E+00)[a]	−9.1 (2.92E+00)[a]

[a] The value of t with 49 degrees of freedom is significant at $\alpha = 0.05$ by two-tailed test.
[b] means RCBBO-EP is worse than its competitor.

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 and the diversity of the population, the mutation operator is integrated into the habitat mutation. In addition, three mutation operators (i.e., Gaussian mutation, Cauchy mutation, and Lévy mutation), which have been widely used in EP, are chosen to verify the enhancement of the exploration ability and the improvement of the diversity of the population.

To evaluate the performance of our RCBBO approach, 27 benchmark functions of a wide range dimensions and diverse complexities are selected from the literature. The results are compared with BBO, which does not use the mutation operators in EP. Experimental results show that (i) our proposed real-coded BBO approach can obtain good performance for the global continuous optimization problems, and (ii) the mutation operator is able to improve the exploration ability and the diversity 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. Moreover, RCBBO-EP is compared with MSEP, FEP, and CEP. The results indicate that RCBBO-EP is better than, or highly competitive to, these three approaches.

Since the habitat migration operator described in Algorithm 1 can utilize the information of current population efficiently, it can be hybrid with other EAs to design more robust hybrid meta-heuristic algorithms. Our future will consist on hybridizing BBO with other EAs for the global optimization problems.

Acknowledgements

The authors sincerely thank Dr. D. Simon for his constructive comments on our work. They are also grateful to the anonymous reviewers for their suggestions on this paper.

References

[1] X. Yao, Y. Liu, G. Lin, Evolutionary programming made faster, IEEE Transactions on Evolutionary Computation 3 (1999) 82–102.
[2] T. Bäck, U. Hammel, H-P. Schwefel, Evolutionary computation: comments on the history and current state, IEEE Transactions on Evolutionary Computation 1 (1997) 3–17.
[3] K. Deep, M. Thakur, A new mutation operator for real coded genetic algorithms, Applied Mathematics and Computation 193 (2007) 211–230.
[4] X. Yao, Y. Liu, Fast evolution strategies, Control and Cybernetics 26 (1997) 467–496.
[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 (2006) 281–295.
[6] R. Storn, K. Price, Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, Journal of Global Optimization 11 (1997) 341–359.
[7] D. Simon, Biogeography-based optimization, IEEE Transactions on Evolutionary Computation 12 (2008) 702–713.
[8] D. Whitley, S. Rana, R.B. Heckendorn, The island model genetic algorithm: on separability, population size and convergence, Journal of Computing and Information Technology 7 (1998) 33–47.

Author's personal copy

[9] D. Simon, The Matlab code of biogeography-based optimization, 2008. Available at: <http://academic.csuohio.edu/simond/bbo/>.
[10] C.R. Reeves, Genetic algorithms and neighbourhood search, Selected Papers from AISB Workshop on Evolutionary Computing LNCS 865 (1994) 115–130.
[11] Z. Yang, J. He, X. Yao, Making a difference to differential evolution, Advances in Metaheuristics for Hard Optimization (2007) 397–414.
[12] C.Y. Lee, X. Yao, Evolutionary programming using mutations based on the Lévy probability distribution, IEEE Transactions on Evolutionary Computation 8 (2004) 1–13.
[13] H. Narihisa, T. Taniguchi, M. Ohta, et al., Evolutionary programming with exponential mutation, in: Proceedings of Artificial Intelligence and Soft Computing, 2005, pp. 1–6.
[14] 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 Fifth IEEE International Conference on Cognitive Informatics (ICCI2006), 2006, pp. 675–679.
[15] H. Dong, J. He, H. Huang, W. Hou, Evolutionary programming using a mixed mutation strategy, Information Sciences 177 (2007) 312–327.
[16] W. Feller, An Introduction to Probability Theory and its Applications, third ed., vol. 2, Wiley, New York, 1971.
[17] B. Gnedenko, A. Kolmogorov, Limit Distribution for Sums of Independent Random Variables, Addition-Wesley, Cambridge, MA, 1954.
[18] I. Ono, H. Kita, S. Kobayashi, Real-coded genetic algorithm using the unimodal normal distribution crossover, Advances in Evolutionary Computing (2003) 213–237 (Chapter A).
[19] K. Deb, J. Dhiraj, A. Ashish, Real coded evolutionary algorithms with parent centric recombination, in: Proceedings of the 2002 Congress on Evolutionary Computation, vol. 1, 2002, pp. 61–66.
[20] J. Rökköen, S. Kukkonen, K. Price, Real-parameter optimization with differential evolution, in: Proceedings of the 2005 IEEE Congress On Evolutionary Computation CEC2005, 2005, pp. 506–513.
[21] 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 (2006) 646–657.
[22] W. Gong, Z. Cai, L. Jiang, Enhancing the performance of differential evolution using orthogonal design method, Applied Mathematics and Computation 206 (2008) 56–69.
[23] P.N. Suganthan, N. Hansen, J.J. Liang, K. Deb, Y.P. Chen, A. Auger, S. Tiwari, Problem definitions and evaluation criteria for the CEC2005 special session on real-parameter optimization, 2005. Available at: <http://www.ntu.edu.sg/home/EPNSugan>.
[24] C.H. Goulden, Methods of Statistical Analysis, second ed., Wiley, New York, 1956. pp. 50–55.